UNIVERSITY OF OSLO
Department of informatics

Semantic Technologies and Publish/Subscribe in Support of Distributed Decision Making

Master thesis
60 credits

Joachim Reitan

04.05 2009

Abstract

Semantic technologies are a family of specifications and accompanying tools that aim to raise the level of data processing from the syntactic to the semantic. The research community expects this to bring important advantages to information integration and sharing, culminating with the realisation of a semantic web.

Publish/subscribe is a paradigm for asynchronous exchange of messages between anonymous and loosely coupled clients. The routing of messages is based on the expressed information need of the receiving clients.

This thesis outlines the design and properties of a publish/subscribe notification service that uses semantic technologies for selective filtering and routing of notifications. In addition, it discusses how the proposed solution might be applied to a specific domain. Network Centric Warfare (NCW) is chosen for application domain. NCW is a military doctrine with the ambition of using information technology to reform the way military forces are organised and operate.

Acknowledgements

First of all, I would like to thank my supervisor, Professor Josef Noll at UniK. You have been very valuable in leading me onto the right track, pointing out things I have overlooked and, not least, stopped me from getting lost in every possible digression along the way.

Then I would like to thank my employer, Forsvarets Forskningsinstitutt, and in particular Frank Steder and Rune Stensrud for accepting this long period of partial absence.

Last, but above all, I would like to thank Ellen Skjold Kvåle, my dear girlfriend. Thanks for putting up with me in this period — and for the proof reading, by the way. Without you, this thesis would not have been the same. Neither would I, come to think of it.

Contents

Chapter 1

Introduction

1.1 Motivation

This thesis is submitted as part of the interdisciplinary study programme *IT—Språk, Logikk og Psykologi* (IT-SLP) [1]. Within the programme, my field of study is decision making and decision support. Thus, when I first heard about semantic technologies, I found it very appealing as a possible way of structuring and processing information to aid decision makers. Rule based systems have long been an important part of the family of decision support systems, and semantic technologies may bring some of that heritage further by providing richer information models through ontologies and inference capabilities. Therefore, I wanted to explore this area in the work with my masters thesis.

For me, purposeful application of technology is more interesting than the technology in itself. Having a background as an army officer, it seemed natural to study a potential application of technology to the military domain. A theoretical framework for the application presents itself in the concept of Network Centric Warfare, which has attracted much attention in recent years. One of the primary goals of this doctrine is to improve decision making by means of more effective and efficient information systems. The military domain as an application area has become even more relevant to me professionally since I started working at the Norwegian Defence Research Establishment [2] in the course of my work with the thesis.

1.2 Background

With the general advancement of information technology over the past two decades, new ideas of how to organise, operate, coordinate and exercise command and control of military organisations have evolved. These ideas

[1] (IT—Language, logics and Psychology)
[2] Forsvarets Forskningsinstitutt (FFI)

have been distilled into a doctrine which is most commonly known by the name *Network Centric Warfare*. The break from traditional organisation of military organisations is quite radical: Static, hierarchical structures with long planning cycles, centralised command and control and non-interoperable information systems are meant to give way to a more flexible, responsive and adaptive force. So far, most of the ideas remain a vision for the future. Experiments and partial realisations of the doctrine have been carried out. The full *revolution of military affairs*, however has yet to become a reality. None the less, the vision has been received as more than a utopian dream: The Norwegian Armed Forces, as well as NATO and other nations, have decided to evolve towards becoming network centric[30, 29].

Publish/subscribe is a communication paradigm that aims to provide loose coupling and asynchronous message exchange between heterogeneous clients in a network. Above all, the purpose of the publish/subscribe interaction pattern is to get the right information to the right point based on the expressed *information need* rather than identity or location. 'Point' can be read as *person*, but it can equally well be read as *machine* or *autonomous agent*. A variety of technological approaches and implementations of the core principles have been put forward, and publish/subscribe has an active research community working on further development.

Publish/subscribe is an exchange pattern suited for highly distributed and heterogeneous systems with complex and unanticipated interactions. In that capacity, it appears to be an interesting candidate technology for the information infrastructure of a networked military force.

Semantic technologies is a family of specifications and tools that spring out of the semantic web vision originally presented by Tim Berners-Lee[75]. After years of development, the field can still be characterised as emerging and promising. Practical applications exist, but so far, the vision has not been fully realised. In more bounded and controlled environments than the world wide web, however, it might be easier to exploit the powerful features of semantic technologies.

As a point of departure, I have defined an application area and two technological paradigms with characteristics that appear to coincide with the requirements of the domain. A general description of this study is an investigation into the possible application of publish/subscribe and semantic technologies to Network Centric Warfare.

1.3 Problem statement

Two main questions are discussed in this thesis:

1. How can information routing and event composition in a publish/subscribe infrastructure be enhanced with semantic technologies?

2. How can the two technologies together be used for selective information dissemination in a network centric military force?

The first question will be treated on a conceptual level more than on a detailed implementation level. The second question will be examined along two paths: One is by general discussions about the characteristics and requirements of Network Centric Warfare compared to the capabilities offered by the two technologies. The other is by describing examples on how these technologies might be applied.

1.4 Thesis overview

Chapter 2, The military domain will give an introduction to the Network Centric Warfare doctrine, goals and theoretical concepts. From that, the chapter will outline what kind of capabilities a supporting information infrastructure must provide and what constraints it must operate within. A background scenario for examples through the following chapter will be described.

Chapter 3, Technology evaluation will introduce the relevant technologies. I assume that the reader has a working knowledge of semantic technologies, but knows little about publish/subscribe. Therefore, the principles of publish/subscribe will be explained, whilst the section on semantic technologies will be very brief. I will discuss how the technologies match the requirements of Network Centric Warfare, and how they may fit and leverage one another.

Chapter 4, Semantic technology enhancing publish/subscribe is the main chapter. The first part of the chapter is concerned with semantic matching and routing of messages in a publish/subscribe notification service. The second part is concerned with composition of higher level event notifications from more primitive ones. The chapter outlines a high-level design of a way to use semantic technologies in a publish/subscribe notification broker. In addition, it provides examples of how the general ideas can be applied.

Chapter 5, Discussion reviews related work and relates my work to it. The chapter briefly presents some open issues that cannot be fully covered here, but still cannot be ignored.

Chapter 6, Conclusions summarises the thesis and points to possible future work.

Chapter 2

The military domain

The application domain chosen for this thesis is the military domain, and more specifically, the emerging doctrine of what is become known as *network centric warfare*(NCW). The basic ideas of network centric warfare are both inspired by- and an inspiration for advancements in information technology, including the approaches discussed in this thesis. This chapter presents some of the key demands and constraints that a network-enabling information system must meet. To show why, the chapter starts with an introduction to some of the background and fundamental concepts of network centric warfare.

2.1 Network centric warfare

Military forces of the world have traditionally been organised in strict hierarchies, and they have operated accordingly. The chains of command have been vertical, with information flowing along the same lines: either 'up' or 'down'. Along with that, the supporting information systems have generally been of 'stove-pipe' design with proprietary representation formats, tight coupling between components, implicit semantics hardcoded in the application logic etc. [66]. Together, this has resulted in relatively inflexible organisations with limited ability to ad-hoc interoperation between subunits across the hierarchy. Another typical consequence is that information moves slowly, resulting in slow decision making and long response times.

Towards the end of the nineteen nineties, technological advancements gave rise to the idea that it should, at least in principle, be possible to overcome some of the limitations and make military forces more nimble and responsive. The book *Network Centric Warfare*[12] established the term as well as the fundamental ideas of the doctrine. Since then, the ideas have evolved somewhat, but the main points remain valid. The identifying tag 'network centric warfare' itself, however, has been complemented with other terms that describe more or less the same thing. Network Centric Warfare

was the original name used in the US, but to reflect that modern military operations include other activities than war, *network-based operations* has been forwarded. The official UK term is *Network Enabled Capabilities* (NEC) [76], and NATO has adopted a similar name, calling it *NATO Network Enabled Capabilities* (NNEC). [55]. In Norway, the corresponding concept is *Nettverksbasert Forsvar* (NbF), which translates to *Network Based Defence* [30]. Without engaging in a discussion about which name is more descriptive, I will mainly use *Network Centric Warfare* as that is the most established and well-known term.

2.1.1 Intent and ambition of Network Centric Warfare

Ultimately, the goal of network centric warfare is to carry out more effective operations. Effective operations was originally about increased combat power in a traditional war scenario [12]. Today, *operations* should be interpreted in a wider sense, and includes all kinds of military means, from peacekeeping and policing missions to full-scale warfare [14, 30]. It also includes activities that strictly speaking are not of a military nature, but none the less important for mission success, like providing the local population with medical services. The concept of *effects-based operations* captures this inclusive interpretation with its emphasis on employing both military and nonmilitary capabilities to obtain the desired strategic outcome [19, 71]. This leads to another point in the promotion of network centric warfare: A networked force does not only have the potential to be more lethal and destructive than an old-style platform-based force. By virtue of being more adaptive, it can apply measures that achieve the objective with less collateral damage to civilian life and property [30].

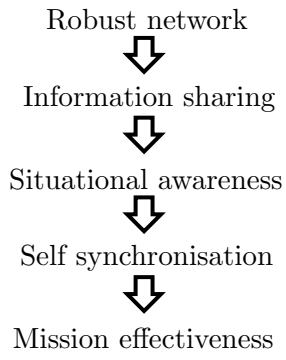


Figure 2.1: Simplified cause–effect chain of Network Centric Warfare.

The driving logic behind Network Centric Warfare is simple (figure 2.1): A robust network makes increased information sharing possible. Information sharing and collaboration enhances the shared situational awareness. Shared situational awareness makes self synchronisation possible, and together this will make mission effectiveness increase significantly. To see how, the next sections will examine the concepts of *shared situational awareness* and *self synchronisation*.

2.1.2 Shared situational awareness

Situational awareness has been defined as "the perception of the elements in the environment within a volume of time and space and the projection of their status in the near future" [26]. Put another way: situational awareness is about knowing what is in your environment and understanding how it will affect you and your objectives. In the extension of this, *shared* situational awareness involves more than one individual or actor. It can be a case of two or more actors cooperating to reach a common objective, such as defeating an opposing force, or it can be a case of more indirect influence of another participant's sphere of interest, such as passing through somebody else's area of observation. In any case, each actor should have a good idea of the intents and activities of the others It's not necessary, or even desirable, to have the exact same awareness as the others. Rather, the awareness of any one actor should be interwoven with the awareness of the collaborators, with sufficient overlap to interconnect the actors [72]. That means there are things both need to know and understand in the same way, there are things A needs to know that are of no interest to B and vice versa. Seen as a global whole, however, the whole picture needs to be covered , and each of the actors needs to be aware of everything relevant to himself as an individual operator, and as a co-operator.

Network centric warfare theorists distinguish between the *physical domain*, the *information domain* and the *cognitive domain* [13]. Situational awareness is part of the cognitive domain, meaning that it is inside peoples heads. Having sophisticated information systems does not automatically mean having good situational awareness, and having information available in technical systems certainly is not the same as having situational awareness. Rather, well designed information systems are a prerequisite to situational awareness — at least in environments as complex as the ones military organisations are designed to handle. In such circumstances, 'looking around' won't be enough to get the full picture, and information systems become essential to maintain adequate awareness.

Shared situational awareness of high quality is the key to self synchronisation between the individual elements in a networked force. The next section will describe what *self synchronisation* is and why it is a central construct in network centric warfare.

2.1.3 Self synchronisation

Self synchronisation is basically a new twist to the old concept of *synchronisation* in military doctrines. To get an intuitive impression of what it is, one can imagine cogwheels smoothly rotating and engaging one another. This physical form of synchronisation is a useful metaphor for the more abstract notion. In military theory, synchronisation is a matter of coordinating and

orchestrating the activities of different units in time and space. The activities in question are typically fire and maneuver, but also logistics and other support activities, as well as any other effects based operation [78]. The purpose is to obtain maximum effect from the resources invested, and not just in a linear, additive manner: Two operations that would be relatively ineffective in isolation, are expected to be immensely more effective when properly synchronised. For example: Close air support[1] and an infantry assault against a well prepared opponent have little effect if the latter follows hours after the former. But with more careful synchronisation, it is a very effective and efficient combination. This is *synergy* in military operations, and it is what makes synchronisation work as a so-called *force multiplier*.

From this, it becomes evident that any method or mechanism capable of improving synchronisation would be of great value to military forces. And in most existing organisations, there is considerable room for improvement.

The traditional way of leading and synchronising military units is influenced by the industrial age way of organising work of any kind. Decomposition, specialisation, hierarchy, optimisation, deconfliction, centralised planning, and decentralised execution are important principles [14]. In practice, this comes about as division of responsibility and separation of actions in time and space — mechanisms that are clear and simple enough to be managed with the (simple) means available. The need to avoid accidental targeting of friendly units and other kinds of unwelcome interference, necessarily leads to a rather cautious practice. Due to the limitations of communications and information technology, most of the synchronisation has to be preplanned. Deviating from the plan to seize a sudden opportunity, or counter an unanticipated threat, becomes difficult and risky. Certainly, there is room for improvisation and adaptation within the bounds of the plan, but anything that runs counter to the plan, and anything that might interfer with the actions of neighbouring units, will have to handled through the hierarchy. This results in an inflexible mode of operation with long response times, and the cycle of planning and execution will broadly speaking follow its own pace whatever happens on the ground.

This way of executing command and control is not just a matter of organisation and doctrine. It's been thoroughly embedded in the technical systems. In most cases, a system, like an anti-aircraft missile system, is self contained with all the components it needs. For the anti-aircraft system, that would be equipment to detect and track hostile aircraft, evaluate the threat and prioritise between possible targets, issue commands, launch missiles and evaluate the impact. This includes radars, communications and information systems, as well as missiles and launchers. The problem with this approach is not that such an integral system has a complete range of resources that it can rely on — the problem is that it *has to* rely on them: Not only is

[1] Fire from aircraft against targets on the ground.

the system self contained — it is often technically sealed off from other systems. Moreover, it can even be sealed off from the same kind of system in a different branch of the hierarchy: If the radar of, say, a Navy vessel covers an area that the anti-aircraft system cannot cover itself, the two typically cannot exchange target data. Partly, this isolation is due to the two not having direct communications, but even if they had, the information systems could not be expected to work together. Data models, semantics and formats, communication protocols and application logic are all made for a single purpose within a single system, and interoperability is not a primary concern in the design.

Self synchronisation refers to the direct coordination and orchestration of effort that occurs between different units or elements. They can be on the same or on totally different levels in the hierarchy, and the synchronisation can be more or less spontaneous and driven by events. It appears either in the pursuit of a common goal, or in the pursuit of different, non-conflicting goals that still has some influence on one another. One example can be a convoy of vehicles that learns about a possible threat along the route they were meant to follow. To reach their destination safely, they will have to follow an alternative route, passing through the sector of another unit. Naturally, that would have to be coordinated. In the case of an international operation, a fairly simple request like that might engage a whole lot of the chain of command and lead to considerable delay — at least under the traditional paradigm for command and control. Directing the request straight to the local commander, and merely informing anyone else that needs to know, would allow for quicker and more adaptive execution.

An important characteristic of self synchronised units is that they can operate in the absence of central command and control. At first sight, this can look like a replacement for the old structure, but that is not what it is meant to be. Instead of replacing the traditional chain of command, self synchronisation is supposed to complement it[14]. The commanding officer should be clear about his *intention* with a mission, and leave the details of the execution to his subordinates. Then, each participating subcommander will carry out his mission guided by that intention, general rules of engagement, his own knowledge and experience, and trust in his fellow officers. Aided by the right information systems, he will synchronise his actions with whoever it becomes necessary to deal with, engaging higher levels in the hierarchy only to the extent that it makes sense.

2.2 Requirements for the supporting information systems

Essentially, network centric warfare is a philosophy grounded in the promises of information technology. Organisational issues aside, the realisation of

Network Centric Warfare rests heavily on technology — finding it and using it for maximum effect. "Finding" is in fact an appropriate word since military research organisations have ceded the technological lead they once held to civilian institutions. Now, it is more a question of appropriating and utilising existing technology than of inventing new. To do that, a clear idea of the purpose is necessary, and then the requirements and constraints for the supporting solutions can be worked out. Based on the identified requirements, candidate technologies can be evaluated and selected. This section will go into some of the functional and non-functional requirements that are posed by Network Centric Warfare, as well as other considerations that seem relevant. It's not meant to be an exhaustive review of every possible aspect of the Network Centric Warfare realisation, but a selection of issues that bear on the technologies under discussion.

2.2.1 Decision support — building a common operational picture

Above all, Information in Network Centric Warfare is there to support decision-making, as opposed to learning, evaluation, research, entertainment, socialising or any other purpose that may be important in other contexts. Decisions are translated to action through command, control and coordination. The *OODA-loop* (figure 2.2) of John Boyd has been established as the standard model of command and control in Network Centric Warfare[32] — as well as in any military context. It describes decision making as a cyclic process: Observe—Orient—Decide—Act. The decision cycle, and the accompanying decision aids, have focused on planning. Planning is and will remain important, but with the increasingly dynamic and agile modes of operation that come with Network Centric Warfare, more emphasis will be put on handling the evolving situation: Unanticipated events and sudden changes need to be detected and comprehended. The plan must be updated accordingly, or downright abandoned, and new commands and coordinating messages must be issued. Also, the OODA planning cycle should speed up: The time it takes to complete a cycle should be shortened. Together, this should result in a faster pace of operation that will leave the adversary disoriented and unable to keep up with events.

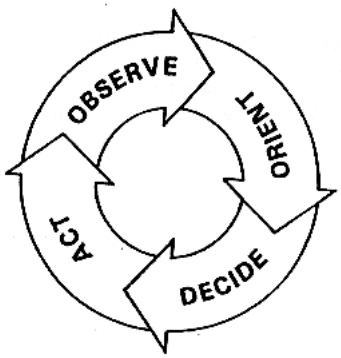


Figure 2.2: The OODA model of a cyclic command and control process. From [9].

This ability to act rapidly and well-adapted springs out of good situational awareness. The vehicle to achieve situational awareness is a *situational*

picture[31] that is complete, up to date and reliable. The situational picture is the representation of the physical domain that the information system presents to its users. Note that the term *picture* should be interpreted broadly as all kinds of text, images, sounds etc. that are informative to the end user — presentation format is of minor significance in this context. Also, the same way situational awareness is shared and distributed among actors, the global situational picture is a collection of smaller pictures that are consistent with one another and build on the same data.

The responsibility of the information system is to assist in finding, selecting, sharing and making sense of all the data. In order to relieve the humans in the process, picture building should ideally be automatic and self organising. In a distributed system, that necessarily means some degree of machine to machine interaction. Autonomous agents will take part in the picture generation, acting on behalf of the human agents, and interacting with other agents[40]. Each participating agent will consequently be both a provider and a consumer of data and fractions of pictures. Judging what information belongs in a picture, what it means in the current situation, and what is more or less important, is in a sense decision making in its own right. So to rephrase the old adage: The information system should provide the right information to the right decision-maker — man or machine — at the right time.

Ironically, even though an important point of Network Centric Warfare is to empower the lower level commanders, a networked environment allows for — and maybe even invites — the opposite: Detailed feeds of real time information to central headquarters makes high-level, or even political, micromanagement of tactical execution possible to an unprecedented degree.

Failure to present a suitable picture will result in all the well-known problems of too little, too much or wrong information. In a networked organisation with an abundance of data available, information overload may well be the most pressing problem. All the right information is there — buried and hidden in all the other information. Technology that can help recognise, filter, integrate and present the right pieces of information will not only be valuable, but indispensable. The information system should both help in making decisions and in resisting the temptation to interfere in judgements that are best left to others. A rudimentary interpretation of Network Centric Warfare is that everyone should have access to all information. A more refined interpretation is that everyone should have exactly the information they need in the current situation — no more and no less. The question is: What exactly is the right information? And how are the agents to know?

Information consumers need a way to specify their needs precisely and unambiguously. Then, the picture production mechanisms should be generic enough to do the job based on the specifications, and not based on hard-coded business logic. Hard coding specifications and rules, which is the conventional way of implementing applications, does not provide for the desired

adaptability to unforeseen situations. Certainly, one can set parameters and preferences, but in the end, any program is made with a specific use in mind. And changing it is a laborious and error-prone process. With more generic agents, the business logic can be taken out of the program and specified in separate models. The models can then be shared, reused and modified much easier than the program code.

2.2.2 Information infrastructure

The information system giving rise to Network Centric Warfare will never be materialised as one monolithic system. Rather, it can be expected to take shape of a *system of systems*. As a general abstraction, the elements in the system of systems can be divided in two: The applications that deliver the actual functionality of value to end users, and the *information infrastructure*[40] that binds the various applications together to create a coherent whole. In the original Network Centric Warfare parlance, the information infrastructure is called *the grid*[13]. The grid is conceived as a decentralised network connecting all the individual actors and elements — including NATO central command, nuclear submarines and single riflemen alike. It should offer the services necessary for collection, integration, distribution and sharing of information between the connected nodes. The Norwegian concept for Network Based Defence uses the term *information infrastructure*, or in short *infostructure*. Because "information infrastructure" is more thoroughly described and analysed than "the grid", I will mainly refer to the Norwegian term.

2.2.3 Flexibility

The envisaged system of systems is not at all going to be a fixed and stable setup of components. Military forces will be assembled and re-assembled for widely different missions in different environments. Actors should be allowed to connect and disconnect in a plug-and-play manner, and they frequently will. Consequently, the supporting information systems need to be very flexible.

In the industrial age warfare, military organisations were to a large extent tailored for specific scenarios against a clearly identified adversary with well-known capabilities and modes of operation. Organisation, plans and standard operational procedures were rather static, and made it predictable who would collaborate with whom, and in what way. The supporting information systems are characterised by rigid processes with both information sources and data flow predefined[40]. Greater flexibility is one of the promises of Network Centric Warfare. In the information age warfare, universal interoperability is essential. Collaboration will often be ad-hoc and short term — and it can be with practically anyone: National as well as inter- and extra-alliance. Small,

low-level units may be assembled at short notice, and will be expected to work without long-lasting preparation. Even civilian actors may be part of the "extended force". A trend in military logistics is to outsource specialised operations to civilian contractors. For instance, advanced maintenance of the Joint Strike Fighter will be done by the manufacturer.

This is not to say that organisational structures will be broken up all together. Rather, there will be established sorts of virtual organisations alongside the more permanent ones. And more importantly: Collaboration will occur between actors that have never been explicitly told to collaborate. Self synchronisation will need to happen across organisational structures. Along with this, workflow will not necessarily be predefined. Instead, activities will be driven by events, and command and control will be more about handling events than executing plans.

The different systems connected to the infrastructure are likely to be heterogeneous technically, procedurally and in every way. Heterogeneity on the web may be even bigger, but much of the same problems apply. One-to-one integration means $N(N-1)/2$ relations in the worst case, and for any realistic N, that becomes prohibitively complex and expensive. The apparent solution is to define common standards for data exchange that all must comply with. This includes common data formats, and furthermore; interacting components need a common understanding of the *meaning* of information[40].

Handling dynamic interaction comes down to being able to do two things: Finding the right information, wherever it is, And conversely; pushing information to anyone who needs it. Information should be an asset offered to anyone and accepted from anyone based on the need at any point in time — not based on identity or affiliation of the actors. One typical case is that anything with a sensor capability should be able to feed information to any shooter.

2.2.4 Disadvantaged grid

Information sharing in the network will only be as effective as the underlying communication carriers permit, and that is an issue of practical concern when military communications are involved. Out in the field or at sea, wireless networks will be the norm, and they can be expected to have severe limitations such as low bandwidth, high delay, high error rates and frequent disconnections [45]. *Disadvantaged grid* is the commonly used name for this kind of networks. Table 2.1 shows an overview of maximum and expected bandwidth for satellite and different radio communications.

High band Ultra-high frequency (UHF) is used for communication between vehicles and a local headquarter (HQ), which is a temporary installation in the area of operation. The Standard NATO Agreement (STANAG) 5066 and 4583 are standards for high frequency (HF) radio communication between a

Link Type	**Max bandwidth (Kbit/s)**	**Expected bandwidth (Kbit/s)**
High band UHF	300.0	300.0
Sub Network Relay	64	64
Personal Role Radio	38.4	38.4
STANAG 5066 / STANAG 4538	9.6	0.15 – 9.6
Satellite (unstabilised antenna)	2.4	2.4
Multi Role Radio / Light Field Radio	2.4	1.0

Table 2.1: Maximum and expected bandwidth for communications on the tactical level. From[45]

vessel and the central, permanent HQ. Sub Network Relay is used between vessels in a group, and one of the vessels acts as a gateway to the HQ via the above mentioned HF radio or satellite. Satellite communication can also connect the central HQ, a local HQ and a vehicle. Multi Role Radio is used between vehicles, Personal Role Radio is used between soldiers on the ground and their squad leader, and the squad leader has a Light Field Radio to communicate with vehicles and the local HQ. All kinds of traffic — data, video, speech etc. — is supposed to be supported by the same carriers.

Exactly who uses what communication means is not the main point here. More important, though, is the fact that a system designer cannot pick and choose communication means freely. The radios on offer are the ones available in foreseeable future, and they generally offer a fair compromise between range, security, bandwidth, weight battery consumption and other constraints. Moreover: The physical connections are structured hierarchically. The hierarchy may or may not correspond to the regular organisational structure, but it generally will. When Network Centric Warfare promises to connect anyone to anyone, that is only a virtual connection. For example, if a soldier on the ground needs to communicate with a ship, there is most likely no radio link that connects them directly. The signals will have to be transmitted through all the intermediate steps in the hierarchy, consuming and competing for resources along the way.

To put the figures from table 2.1 in perspective, table 2.2 shows a similar table of some better known civilian communication carriers. Throughput requirements for different applications should also be illuminating. In [65], they are estimated as follows:

- Microbrowsing (e.g., WAP): 8 – 32 kbps

Link Type	**Peak network downlink speed (Kbit/s)**	**Average user throughputs for file downloads (Kbit/s)**
GPRS	115	30 – 40
EDGE	473	100 – 130
UMTS WCDMA	2 000	220 – 320
UMTS HSDPA (Telenor "Turbo-3G“)	14 000	550 – 1 100

Table 2.2: Maximum and expected bandwidth for some civilian wireless communications. From [65]

- Multimedia messaging: 8 – 64 kbps
- Video telephony: 64 – 384 kbps
- General purpose Web browsing: 32 – 384 kbps
- Enterprise applications, including email, database access, virtual private networking: 32 – 384 kbps
- Video and audio streaming: 32 – 384 kbps

The exact figures may be debated, but they illustrate the idea: Bandwidth is indeed a scarce commodity in military communications. This is very different from the trend in most civilian enterprise settings where network resources are abundant, cheap and constantly increasing.

The take-home message is stated clearly in [37, p 17]: "In a DG, the network is the limiting factor, and not the processing capacities of the nodes[. . .]“. In practical terms, almost any amount of processing is acceptable to reduce network traffic. Of course, processing capacities are not unlimited either. Personal digital assistants (PDA) is an example of a kind of device that individual soldiers will be equipped with. Central nodes in the infrastructure will hopefully be more powerful, but at some point; processing becomes a bottleneck. Still, the limitations of the tactical communication carriers are severe and can be expected to remain so. Hence, a practical rule of thumb for evaluating candidate system designs for the military domain will be: A computationally expensive solution which consumes less network resources will generally be better than a computationally economical one that consumes more.

Two additional considerations have been pointed out by Hafsøe et al.[37]. First, they recommend that to achieve an acceptable throughput (more than 50% of the bandwidth), packet size should be at least 2-5kB. Second, changes

in sending direction incurs delays, and should be kept to a minimum. In an environment where continuous updates of an evolving situation is desirable, as the case is with Network Centric Warfare, query based information exchange should be avoided when possible. Each query may not be very expensive, but an information provider that receives frequent queries from numerous receivers will experience considerable traffic — and moreover; uneconomical traffic that consumes disproportionately much of the network resources. Instead, information exchange should be based on pushing information from the provider, and in particular when the receivers request updates of small data sets.

Finally, sudden disconnections will leave actors off line for shorter or longer periods. With wired networks, disconnections can rightly be regarded as an exceptional state, and components in a distributed systems can be designed for relatively stable interconnections. With radio links as communication carriers, however, disconnections will be a fact of life. Therefore, every node must be able to function in isolation for a period, and then catch up when it reconnects.

2.2.5 Security

Security will always be a concern for military applications, and the shift towards Network Centric Warfare does not make it any less so. Greater reliance on common infrastructure means increased vulnerability. With systems designed as isolated silos, each one is relatively easy to control, and failure in one system will not affect other systems. In a more complex and interconnected system, that cannot be trusted to be the case.

Assuming that conventional security mechanisms are effective in stopping intrusion from outside, a system that is open internally must have some additional mechanisms in place. All the actors are on the same net, but should not have access to all information[40]. Security classification and authorisation must be managed, and with dynamic configuration and roles, rights must be updated accordingly.

A networked military force is supposed to operate in hostile environments in more than one sense, technical failures and sheer physical destruction of components is to be expected. Apart from making each component more robust, redundancy in storage, data processing and networking are ways to make the system as a whole more resilient.

2.3 Networked organisations in general

As a small aside, it is worth mentioning that the selection of one specific domain — the military — does not imply that the technological solutions under discussion are unsuited to all other domains. Quite the contrary: The initial model for network centric warfare was commercial organisations taking

advantage of information technology by translating the information superiority into competitive advantage [12]. A war-fighting organisation certainly has to face up to some unique requirements, but not every requirement is so unique that all properties and needs are different from those of a peace-time organisation. In a sense, a networked military force could be regarded as just another kind of networked organisation. Thus, commercial and other networked organisations might also profit from adopting the principles and technologies of network centric warfare.

2.4 Example scenario

An imaginary convoy of vehicles will provide the following discussions with most examples and use cases. A convoy is a number of vehicles travelling together for better protection and safety — or simply for helping a unit stay collected. This way of organising movement is used both on land and sea, but in this thesis, the imaginary convoy will move on land. The convoy is illustrative in that it moves relatively independently through an area, and therefore will need to synchronise its actions with anyone that happens to have an interest or responsibility along the way. Keeping track of friendly forces is an important part of maintaining the situational picture in any operation. And small entities moving around, such as a convoy, is a case that needs to be dealt with. Also, a convoy can run into all sorts of unanticipated situations that require a flexible response and adapted information exchange. Information has to flow both ways: The convoy should inform the outside world about where it is, what it plans to do next, what it observes, problems or emergencies it runs into etc. In return, it needs to be informed about anything that can affect itself in any way; conditions on the roads, friendly activities, observations of enemy activities, appraisals of threat etc. The selection and dissemination of information should be governed by matching the information content with the information need — not by the identity of the actors. Who the convoy needs to communicate and collaborate with becomes a function of the information needs, and not something to be fully identified in advance. Even the internal structure of the convoy is a case of flexible and dynamic organisation. Vehicles from different units, or even nations, will typically come together. For example can lorries from one unit be escorted by armoured vehicles from another unit, and the convoy can have a third unit as destination. In short, the convoy exemplifies the need for open and adaptive information handling in Network Centric Warfare. Also, whilst being common in military operations, this scenario should be easy enough to get hold of for readers without a military background.

Chapter 3

Technology evaluation

With some of the main ambitions and characteristics of Network Centric Warfare laid out, it is time to take a look at candidate technologies for realising the vision. The crucial questions are: What do they offer? And how can different technologies be used to meet the requirements of the application domain? The technology paradigms I have in mind are publish/subscribe and semantic technologies.

Publish/subscribe is basically about distributing information about events from one actor that causes or detects the event to other actors that are interested in learning about it. Comparing publish/subscribe to traditional requests to a server, one might say that the former starts where the latter ends: A request to a server is most often for stored data, whereas a subscription to a publish/subscribe service is in a sense a request for future data — data about events. Publish/subscribe supports flexible event-driven applications across heterogeneous components, and that is above all what makes it interesting for military applications.

Semantic technologies is a set of emerging technologies that receive considerable attention, both within research and for practical applications. Military research institutions were heavily involved in the early stages of development, and it is still an area that holds great promise for military settings. The Norwegian Defence Research Establishment[1] (FFI) currently runs a project to evaluate if, and how, semantic technologies can be utilised in military applications — and in particular how they can support the implementation of Network Based Defence[2][39].

This chapter provides a general introduction to publish/subscribe, and a discussion of what makes it seem suited for information exchange in Network Centric Warfare. Because knowledge about semantic technologies is becoming increasingly widespread, they will only be presented very briefly, and not

[1] Forsvarets Forskningsinstitutt

[2] Network Based Defence (Nettverksbasert Forsvar) is the Norwegian adaptation of Network Centric Warfare

with general, introductory descriptions. Their potential contribution to the realisation of Network Centric Warfare will be discussed. In the final section, the case will be made for combining publish/subscribe with semantic technologies.

3.1 Publish/Subscribe

Publish/subscribe (pub/sub) is more of an abstraction of a communication pattern than a specific technology. There is no one universally accepted standard but several approaches based on somewhat different ideas and terminologies[18, 27, 51, 74]. Publish/subscribe is subject to considerable interest, and a variety of implementations have been realised — both for production and for research purposes. They do, however, adhere to a common set of principles and features that make them stand out as one family of technologies. This section will...

Publish/subscribe is one of several subtypes, or paradigms, within the broad notion of middleware. To set the context, I will start with a brief introduction to middleware.

3.1.1 Middleware

Middleware is software that mediates between an application program and a network. It manages the interaction between disparate applications across the heterogeneous computing platforms[2]. The basic role of middleware is to provide otherwise incompatible applications and components with a common environment to exchange data and services. It stands, as the name suggests, between components in the greater system, and seeks to neutralise the incompatibilities between different operating systems, programming languages, data models and other things that may impede direct interoperation. Middleware is inserted as an extra layer between applications and the network operating system, offering a higher level of abstraction[73]. In addition, and equally important, middleware decouples the components. If the components were to interact directly, each of them would have to interface with all the other components that it needed to interact with. In a heterogeneous environment, that is not feasible, so the need for middleware arises because the components cannot and should not engage one another directly.

Middleware is identified as a key element in the realisation of Network Based Defence[40, 66]. Middleware will probably assume a more active role than merely binding things together. Putting certain functionality into the middleware makes sense since it is a shared resource. 'Put into' can equally well be thought of as finding middleware solutions that innately offer the desired services and properties. A prime candidate for functionality to be included in the middleware is support for platform independent information

exchange and -management. This includes services that have to do with collecting, integrating, aligning and distributing data.

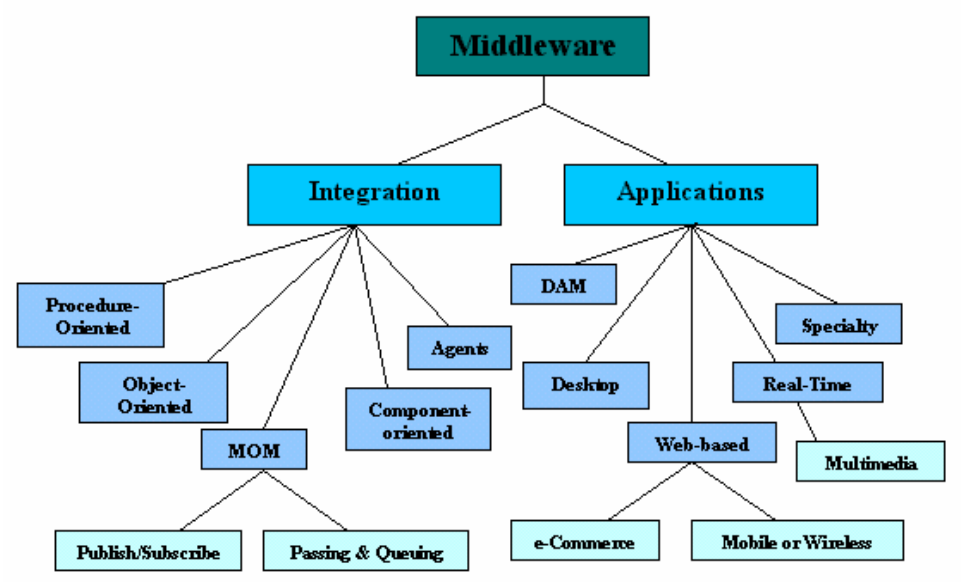


Figure 3.1: Publish/subscribe is one of many different subcategories of middleware. Here is one possible taxonomy, as proposed in [20].

As a subtype of message-oriented middleware (figure 3.1), publish/subscribe contributes to system integration by distributing messages between connected components. The rest of this section will describe how publish/subscribe works in general terms, and some of the advantages this approach has to offer.

3.1.2 events and notifications

At the heart of publish/subscribe is the *event*, and the core responsibility of a publish/subscribe implementation is to distribute data about events between *clients* of the service. The clients are simply software components that connect to the publish/subscribe *notification service*(NS), and any one client can assume the roles of *producer*, *consumer* or both. Any kind of component can be a client of the notification service, be it a user application, a software agent acting independently on behalf of a user, a sensor, a database server, or any other module at the application level.

The *notification* contains information about an event. The same event can be described by several notifications, each providing a different view. Technically, an event is a change of state in a computer system[53]. In practical terms, an event is something that happens in the 'real' world, and which a computer system detects one way or another. *Real* is put in

quotation marks because the question of what is real or not is almost a philosophical one. Clearly, events in the physical world are included, such as a vehicle moving from one place to another. Less clear is the case of inferred or aggregate events. A typical example is a network of hydrophones [3] reporting an event of a submarine passing by. Actually, the hydrophones detected a pattern of noises consistent with the noises expected from a passing submarine. Instead of reporting all the low level noises, the network reported one aggregate event; a submarine passing. That might be true, but it could also be a misinterpretation of something else; like a sea mammal swimming by. Additionally, some events are not directly about anything that actually happened. Instead, they convey the emergence of a relation between objects. For example, when a convoy enters an area, it can also come within the range of fire of some unit. This kind of events will typically be derived from other, more directly observable events. In summary, the term *event* is to be understood in a wide sense.

A *message* is a container for notifications and other kinds of data that is sent between the notification service and the clients, and within the notification service. Messages are transmitted in the network layer, beneath the application layer dealing with notifications. Terminology varies somewhat between different authors and implementations, but this is in keeping with [53].

Unlike the request/response interaction pattern, which is dominant in server/client architectures, the notification producer initiates the exchange — not the consumer. If there is no event to report about, there will be no communication. Thus, all unproductive polling from the client is eliminated. Moreover, the event will be reported the moment it occurs, and not some time after, at the next poll. This leads to a swifter and more responsive information exchange. Together, this makes for an interaction that is inherently event-driven. An event detected in one component can trigger a cascade of events propagating through the system. The resulting process is not controlled by a central structure or scheme. Rather, the behaviour of the system as a whole emerges from the individual messages and reactions they trigger in the different clients that receive them.

Another important feature of publish/subscribe is genuine one-to-many communication. The producer issues a single notification about a specific event. If that notification is of interest to multiple clients, it gets duplicated somewhere along the way, and delivered to the appropriate consumers. Compared to the traffic generated by repeated one-to-one interactions, this can greatly reduce the strain on the producer and its network connection.

[3] Underwater microphones.

3.1.3 Routing of notifications

The flow of notifications through the notification service can be controlled in different ways[18], the simplest one being flooding. Flooding means that any notification is passed on to all clients regardless of their needs. This solution is of little practical interest, and certainly in a military setting. Most events are of interest to only a few other clients, for example the other vehicles in a convoy. So forwarding a notification of mostly local interest to everyone is clearly a waste of resources. First, it burdens the network with unnecessary traffic. Second, it burdens clients with lots of incoming notifications which are essentially noise, but still need to be handled.

Subscriptions are introduced to constrain the flooding of notifications. A subscription is effectively a description of a specific kind of event — like a template. Instead of forwarding an incoming notification to all, it will be forwarded selectively to the clients that have expressed their interest by subscribing to just that kind of event. Presumably, that will be only a small subset of all the clients — both in general, and certainly in a military setting. Because military operations are inherently bound in physical space, it is reasonable to expect that most of the notifications will be exchanged between clients close to one another, and not across the entire organisation.

A client sending a notification to the notification service is said to *publish* the notification — hence the name publish/subscribe. Upon receipt of a notification, the notification service checks to see if it has registered any subscriptions for that kind of event. If so, the message will be forwarded to the interested consumers — they will be *notified*. If not, the message will be dropped. In effect, the notification service acts both as a notification filter and as a router[4] for the notifications by matching them with subscriptions.

Figure 3.2: Clients issue subscriptions for different kinds of events to the notification service.

Figure 3.2 shows the first step of the publish/subscribe interaction. The clients ($C_1 \ldots C_4$) inform the notification service of their desire to receive notifications of different kinds of events ($E_B \ldots E_D$) by issuing different subscriptions. The notification service records the subscriptions in a *routing table* for later lookup when notifications come in. Note that C_1 subscribes to

[4]Routing in the application layer, that is. Routing in lower layers is another matter.

nothing, C_3 subscribes to several event kinds, and C_2 and C_3 both subscribe to E_B.

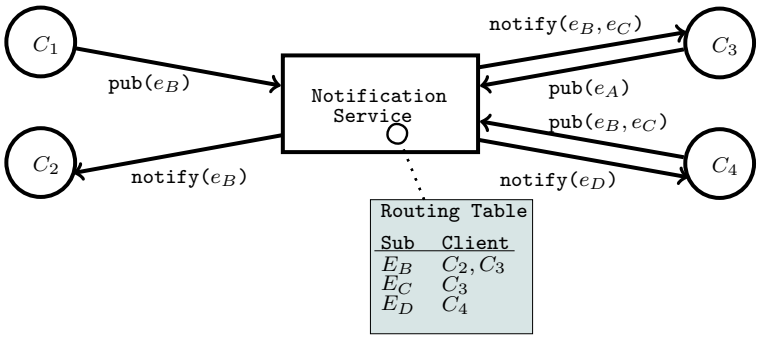


Figure 3.3: Clients publish event notifications of different kinds. The notification service forwards the notifications to clients based on subscriptions registered in the routing table.

The second and third step of the interaction are shown in figure 3.3. First, clients publish zero, one or more notifications about specific events ($e_A \ldots e_D$). For each incoming notification, The notification service finds which clients, if any, has an active subscription for that kind of event registered in the routing table. Then it notifies the respective clients. Note that no client subscribed to events of the kind E_A, so the notification of e_A published by C_3 was dropped. C_3 and C_4 act as both producers and consumers, whereas C_1 and C_2 act only as producer or consumer, respectively.

3.1.4 Classification

Broadly speaking, notification services fall into one of two categories depending on how the matching takes place. The first is *topic-based* matching, where the notification service predefines a set of topics to categorise notifications. In a military setting there could, for example, be a topic for events at a specific location, for a specific type of events, like `airRaidWarning`, or for a specific unit. The message header will declare that it is about one or more topics, and the message will then be considered to match any subscription asking for any or several of those.

The topic-based approach is rather static in that topics need to be predefined. The other approach; *content-based* matching, is more flexible, and also more expressive. As the name suggests, the contents of the notification itself are used for matching. Exactly how depends on the implementation, but the basic scheme is combining key-value pairs. A simple example of a notification could be:

```
(unit = convoy325) AND (action = passBy) AND
(location = junction22-91) AND (directionOfMovement = south).
```

This would match with a subscription for:
`(action = passBy) AND (location = junction22-91)`
`AND (directionOfMovement = south).`

Extensions and refinements of the basic key-value pairs exist, including XML-based content schemes. These allow more detailed and selective matching. Still, they are limited to match on syntax, as we will come back to in section 4.1.

3.1.5 Decoupling

Decoupling of the clients is a prominent feature and benefit of all middleware. Publish/subscribe offers three dimensions of decoupling.

The clients exchanging notifications through a notification service are decoupled in time. That is beneficial because they do not have to be connected at the same time for the exchange to occur. The producer publishes the notification, and if the consumer is off-line at that moment, the notification service will store the message and notify the consumer when it comes back on line.

Within a request/response-pattern, the client that issued the request usually waits for the response to return, blocking further execution of the program in the meantime. Publish/subscribe, on the other hand, works *asynchronously*. All client in the publish/subscribe-paradigm keep their main processes running, and publications and notifications are just small interrupts in the stream. This is referred to as a *non-blocking* mode of operation.

A notification producer does not know who, if any, are interested in the event that has occurred. It simply leaves the filtering and routing to the notification service, which will notify zero, one or more consumers depending on the active subscriptions. In the opposite end, the client that receives the notification does not know where the notification came from. The clients are totally anonymous to one another — at least as far as the notification service is concerned. Should the applications using the notification service need to know who published the notification, that information must be included in the notification itself.

3.1.6 Broker networks

Up to this point, the notification service has been described as one single entity to which all the clients connect — like a centralised server. As a high-level view, that is instructive enough, but in practice, the notification service will be implemented as a network of several *brokers* collaborating to provide the notification service. To a client, that should be fully transparent, and it just registers with a broker it typically finds through a directory service. Clients and brokers alike communicate by exchanging messages. The content

of a message is a subscription or unsubscription, an event notification, or possibly some other administrative notice.

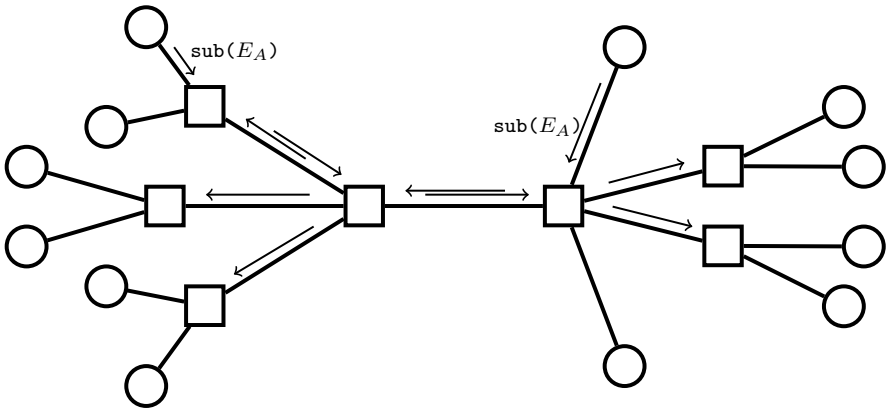


Figure 3.4: Subscriptions are flooded from clients (circles) to all brokers (squares).

The basic routing within the broker network works as follows: Subscriptions are flooded through the entire broker network, each broker passing the subscription on to its immediate neighbours (figure 3.4). Every broker maintains its own routing table, mapping subscription to immediate neighbour node — be it a client or another broker. When receiving a subscription, the broker records the subscription and the neighbour node it came from in the routing table. The result is a trace of the subscription pointing back towards the client that issued it.

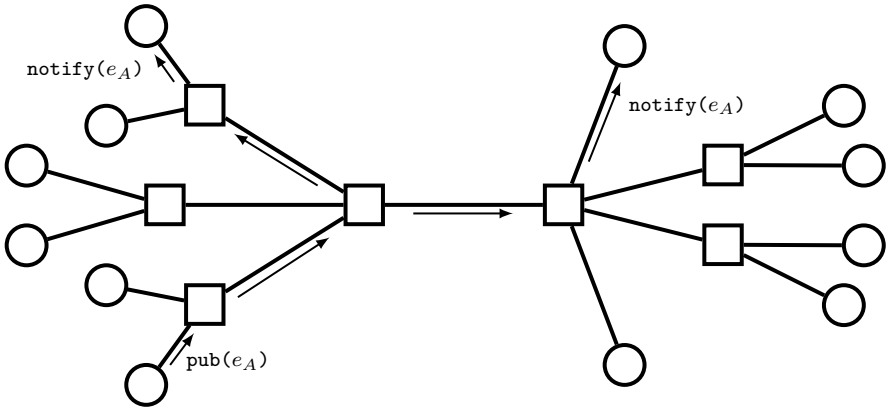


Figure 3.5: Event notification backtracking the trace of the subscriptions.

When an incoming notification matches a subscription, the broker finds which of the neighbours gave it just that subscription in the routing table, and passes the notification there (figure 3.5). This repeats itself one hop at a time.

Unless both the producer and the consumer of a notification are connected to the same broker, None of the involved brokers knows both the producer and the consumer of any given notification. They only keep a record of the immediate neighbours and forward the notification one hop. Eventually, the notification will end up with the client that issued the subscription.

In the case where a notification matches subscriptions from more than one client, the route of that notification will at some point be split. The effect is a one-to-many communication, as opposed to the one-to-one communication inherent in request/reply communication. To minimise network traffic generated by a single publication, the notification service should ensure that the split occurs as late as possible.

The topology of the broker network can, among other, be an acyclic tree, a general graph (possibly cyclic) or some hybrid, where subnets of different topologies are connected. The topology greatly influences how complex the routing becomes. Trees are relatively easy to handle, but cycles introduce certain complications in the dissemination of subscriptions and notifications[49]. On the other hand, cycles offer redundant paths, and by that greater resilience and fault tolerance. For the sake of clarity, however, I will restrict the discussion to acyclic trees for now. The network of brokers as a logical overlay networks on top of the transmission network. Thus, it depends on the services of the underlying network for the propagation of subscriptions and notifications. The topology of the two may or may not be the same.

3.2 Publish/subscribe in support of Network Centric Warfare

After having introduced Network Centric Warfare and publish/subscribe, it is time to examine how the two may fit together. That is; how the mode of operation and characteristics of publish/subscribe may support the needs of Network Centric Warfare.

3.2.1 Flexible information dissemination

Above all, publish/subscribe is about flexible information dissemination. In Network Centric Warfare, interaction between different units, individuals and software components is supposed to be more dynamic and horizontal than in the traditional platform-based defence: An actor cannot always know who in the big grid might need the information he has. Likewise, he cannot always know who may have information that is vital to himself. This may be a matter of more than routine updates of location: In an emergency, a request for supportive fire should reach anyone with available firepower. Shortcutting the hierarchy to connect the two could make a real, operational difference.

Because publish/subscribe does filtering and routing of information based on *content* instead of the identity or organisational attachment of the actors, it seems to be a good fit for these cases. The 'what' of the notification is more important than the 'from where' or 'from whom'.

Military operations are typically not continuous, structured, predictable processes. Sure enough — plans exist, but they rarely provide a precise, not to say complete picture of what will be going on. Hence, operations are much about more or less anticipated events. Event-driven middleware like publish/subscribe supports that in a natural manner. Global predefinition and control of processes and data flow is relinquished. Instead, local judgements, actions and reactions self-synchronise as events unfold. Events trigger new events, and the collective behaviour of the system is allowed to emerge from the individual interactions.

3.2.2 Moderation of resource demands

The restricted bandwidth of military communications dictates efficient and economical communication patterns. The non-existent polling of publish/subscribe nicely serves to reduce unnecessary traffic. Each request may not be very big, but in a highly dynamic battlefield, a lot of polling would go on to keep the situational pictures up to date. In sum, that could amount to conciderable traffic, consuming an important proportion of the available resources — particularly to the peripheral nodes of the grid, where bandwidth is most of a concern.

On the same vain, keeping data in centralised servers is a bad idea in a disadvantaged grid: Requesting and updating data triggers network traffic, and repeated use of the same datum triggers repeated transfers. In contrast, publish/subscribe architectures not only allows, but demands that data be kept locally. That leads to redundant storage, with data about the same events stored in all the notification consumers. Often, such redundant storage is considered a problem — not a benefit. Admittedly, the usual issues of consistency, security and so forth are still valid. None the less, the value of saving network traffic should be emphasised, and presumably outweighs the disadvantages of redundant storage.

In an operational environment, nodes in the grid will come and go, connect and disconnect. The asynchronous nature of publish/subscribe, as well as the decoupling in time, makes that less of a problem than it would be with synchronous request/response interaction. If two actors do not happen to be connected at the same time, they can still exchange data fairly undisturbed as long as they connect once in a while. The notification service will store and forward pending notifications. Also, the aforementioned redundant storage means that they will have data available locally, and not loose everything whilst disconnected.

The more expressive content-based matching schemes of publish/subscribe

also contribute to reduce traffic in that they allow more precise and selective subscriptions to be expressed. So instead of subscribing to a broad topic, of which only a few notifications are really interesting, a client can ask for exactly the information it needs, and no more. The price to pay for expressive matching is heavier processing in the brokers, but with the network as the limiting factor, it is probably worth it.

3.2.3 Example case

As an example case, an observation post (OP) is set up to monitor an area or a point of interest, such as a valley, or a bridge. The OP is not supposed to be seen itself, and will typically hide somewhere with a good view. To make the most out of its observations, the OP should be prewarned about anything that can be expected to enter the area it keeps under surveillance, be it friend or foe.

Supposing the convoy from section 2.4 approaches the area the OP observes. Neither of the two need to know about the existence of the other from the outset. With a well functioning information infrastructure, they will be informed automatically about interesting events if and when the need to know arises. From the point of view of the OP, it would in fact be better to be spared the clutter of details about movement outside its area of interest. Better still if it would not even have to ask for relevant updates, but get them pushed as events occur. So when the convoy reports that it passes a points not far from the area of observation, that report should reach the OP. With a publish/subscribe notification service, this will of course translate into the following sequence:

1. The OP subscribes to notifications about movements of anyone or anything that moves towards it is area and comes closer than the specified point. With a content-based notification service this could be described with the subscription from section 3.1.4:
   ```
   (action = passBy) AND (location = junction22-91)
   AND (directionOfMovement = south).
   ```
   It should not have to specify details such as what or who is approaching, which unit it belongs to etc. In holding with the principles of decoupling, that is just the kind of things a peripheral node like an OP should not have to worry about.

2. The convoy reports its progress by publishing notifications as it moves along; with fixed time intervals, after a certain distance covered and/or upon passing noticeable points. For example:
   ```
   (unit = convoy325) AND (action = passBy)
   AND (location = junction22-91)
   AND (directionOfMovement = south).
   ```
   More information would probably be included, such as the type and

number of vehicles, the approximate speed it travels with and so forth. But for matching the subscription, the above will suffice.

3. Finally, the notification service matches the notification with the subscription and notifies the OP. At the same time, the notification would probably be forwarded to other interested actors, such as the home unit of the convoy.

From the point of view of the convoy, the need for interaction is quite different. The convoy commander does not care about the OP itself, but only about the observations it makes. Most of all, the convoy would want to know about potential threats and obstacles along its itinerary and alternative routes. The OP, not knowing about the distant convoy, will report things as it discovers them. In addition, it may publish "all clear-notifications" once in a while. To the convoy, that sort of notifications will also be interesting as positive confirmations that the road is clear. Knowing that someone watches is reassuring. Knowing who published the notification is less interesting. It could be the OP, a passing helicopter, another convoy etc. To judge how trustworthy the information is, request further detail and such, the identity of the information producer would need to be known. But that is secondary to receiving the right kind of information in the first place; decoupled, anonymously and event-driven.

So far, the notification service has fitted the case quite well. Information is disseminated in the flexible and economical way the situation calls for. But what if the format and syntax of the published notifications differ from that of the subscriptions? For instance, if the convoy reports progress with fixed intervals, it might not publish that it passes junction 22-91, but some arbitrary point just after it. A notification with `(location = 22954 87181)` will not match the subscription. Even a minor deviation as swapping the road numbers of the junction, producing `(location = junction91-22)` instead of `(location = junction22-91)`, would lead to a failed match.

This is a shortcoming of any publish/subscribe implementation based on literal, syntactic matching. Subscriptions can become very detailed and precise with long expressions containing conjunctions and subjunctions of key-value pairs. But the brokers will fail to recognise different expressions of what is semantically the same thing. Given this limitation, it makes sense to turn to semantic technologies for possible enhancements.

3.3 Semantic technologies

Semantic technologies, or semantic web technologies, are intimately related to the semantic web activity of the World Wide Web Consortium (W3C)[11]. Similar, but unrelated approaches exist, such as *topic maps*[57, 8]. However; in order to stay aligned with the ongoing the research efforts at FFI, I

will restrict the scope of this discussion to the W3C recommendations and associated technologies. A graphical representation of their names and interdependencies can be seen in figure 3.6.

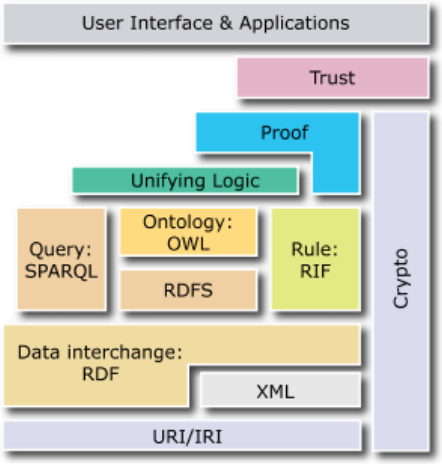


Figure 3.6: The W3C stack.

With the risk of being overly confident on behalf of the semantic web community, I assume that the knowledgeable reader is already familiar with the relevant technologies. Therefore, I choose not to go into general introductions and descriptions of RDF, OWL, description logic, reasoning etc. Instead, I will quote the opening sentence of Grobelnik and Mladenić in [33], which I think captures the essence of semantic technologies:

> We can observe that the focus of modern information systems is moving from 'data-processing' towards 'concept-processing', meaning that the basic unit of processing is less and less is the atomic piece of data and is becoming more a semantic concept which carries an interpretation and exists in a context with other concepts.

The driving vision behind the evolution of semantic technologies has been the semantic web[75, 68]. That does, however, not make the technologies any less applicable to domains outside of the world wide web.

3.4 Semantic technologies in Network Centric Warfare

Hansen et al.[39] have made a preliminary evaluation of how semantic technologies can advance military information infrastructure and applications in direction of Network Based Defence. They point out dynamic and flexible

information sharing as one of the key areas that semantic technologies can address. Information sharing will happen between partners not known at design time, and they can be expected to use heterogeneous and incompatible legacy systems — at least in the foreseeable future.

Integration and management of information is just the thing semantic technologies are designed to handle. In practice, this includes collecting information from different sources, mapping it to common ontologies, merging it and inferring additional, implicit information by automatic reasoning.

The last step of inference corresponds well to the long-standing military need for data fusion[38]. Like inference in semantic technologies, a central goal of fusion is to reveal important relations that cannot be observed directly. It's an extension of integration and alignment of data from several sources in that it seeks to extracts progressively more abstract concepts from the observed signals.

Once merged, aligned and fused, the semantically annotated data can in turn be included in the common operational picture. Thereby, they are readily available for improved decision support. Effective decision support is not only — and perhaps not even primarily — about getting more information, but about getting the right information. Information overflow is a problem that will need to be handled in Network Centric Warfare[81]. Semantic technologies can help make sense of the abundant data, and thereby select and prioritise the most relevant. To relieve the humans in the loop, autonomic agents and services will need to be active in this process of fusion and selection. Machine to machine cooperation of just that kind is part of the semantic web vision.

Achieving effective integration of information depends on the participants adhering to a shared information model. As far as semantic technologies and semantic integration is concerned, that means a common set of ontologies. This is not to be understood as enforcing one model upon all the different information systems that are connected to the system of systems. They may very well use their own internal information models. But there needs to be some common ground for the information sharing; information models and ontologies being part of it. Without some central authority, it is inherently difficult to establish those common ontologies. This is clearly one of the obstacles for the semantic web[56]. In a controlled information sphere like a military organisation, however, standards and regulations may be introduced. Even within such a controlled environment, agreement can be hard to reach, as noted in [39]. They refer to work in the Multilateral Interoperability Programme of NATO, and conclude that "it is not realistic to create one single data model that can capture all relevant information". Still, the programme has managed to standardise information for the operational subdomain. So with an incremental approach, useful models for bounded subdomains should hopefully be within reach. The military domain is big and heterogeneous enough for semantic technologies to be useful, yet small

and controlled enough for their practical application to be within range.

In all, exploitation of semantic technologies in the military domain are in a relatively early stage, but still holds promise worth investigating further. One of the things that needs investigation is how to integrate the 'raw' semantic technologies into the information infrastructure.

3.5 Semantic technologies and publish/subscribe combined

From the discussion so far, both publish/subscribe and semantic technologies are found to be interesting candidates for Network Centric Warfare in their own right. As it turns out, they both address the same basic problem of Network Centric Warfare; Effective information sharing between heterogeneous systems that need to collaborate.

Publish/subscribe has information *dissemination* as its core business, as semantic technologies have information *integration*. These are complimentary functions, where one can profit from the other. Finding ways to combine the two presents itself as an attractive idea.

Publish/subscribe has, as we have touched upon in section 3.2.3 a limited capacity for matching on the semantic content of notifications. Experimental implementations have sought to solve this, but mainly with semantic annotations specially designed for the purpose (see section 5.1.3). Using the — relatively standardised — W3C semantic technologies instead has several benefits. They have a solid formal foundation, and even though semantic technologies are not very mature, there is a large and active research community behind them. So standards and tools can be expected to evolve over time. Besides, using different technical solutions to solve similar problems is poor engineering. If W3C technologies are employed for integration and fusion of information in the first place, they should also be employed in support of publish/subscribe, instead of some customised solution. That will make for a more coherent architecture, which promotes integration, interoperability and reuse.

Semantic technologies are candidates to play a role in the future information infrastructure, but it is not yet entirely clear what role. Effective information integration and -fusion would definitely be valuable capabilities in the information infrastructure, but it depends on some mechanism to make the information come together and be diffused. Information integration and -fusion are in a sense solutions looking for — if not a problem — a suitable framework to function within. Publish/subscribe may be one such framework.

The next chapter will examine a way to combine the two. The objective is on one hand to enhance publish/subscribe with semantic capabilities, and on the other hand to give semantic technologies a place to unleash its potential.

Chapter 4

Semantic technology enhancing publish/subscribe

With publish/subscribe and semantic technologies identified as partial solutions to the same general challenges, it is time to investigate what a possible combination might look like. This chapter offer an outline of a conceptual solution, identifying necessary subcomponents in a single notification broker and their interactions. The two functionalities discussed are semantic routing and construction of composite events.

In addition to technical descriptions, I will go through a few imaginary examples, demonstrating how a notification service enhanced with semantic technology might prove itself useful.

4.1 Semantic matching

The most obvious benefit offered by semantic technologies, is that it allows for filtering on the semantic meaning of the notification, and not only on the syntactic structure. Syntactic matching is basically a matching on strings, possibly enriched with regular expressions to provide some flexibility in spelling. Semantic matching allows this to be taken much further. The basic case is synonyms, which will not be caught by syntactic matching. Semantic technologies makes that straightforward, treating for example the Norwegian concept `EgenKolonne` as equivalent to `FriendlyConvoy`.

Taxonomies is an extension of synonym relations, adding hierarchies of superclass/subclass relations. As an example (figure 4.1), a subscription that requests information about any `MovingFriendlyUnit` will not be matched *syntactically* with notifications about a `FriendlyConvoy`. With the support of an ontology that models a `FriendlyConvoy` as a subclass of `MovingFriendlyUnit`, semantic filtering should recognise the match. In general, a notification about an individual of class C_1 should match subscriptions to individuals of class C_2 if every C_1 is also a C_2. The reverse does not

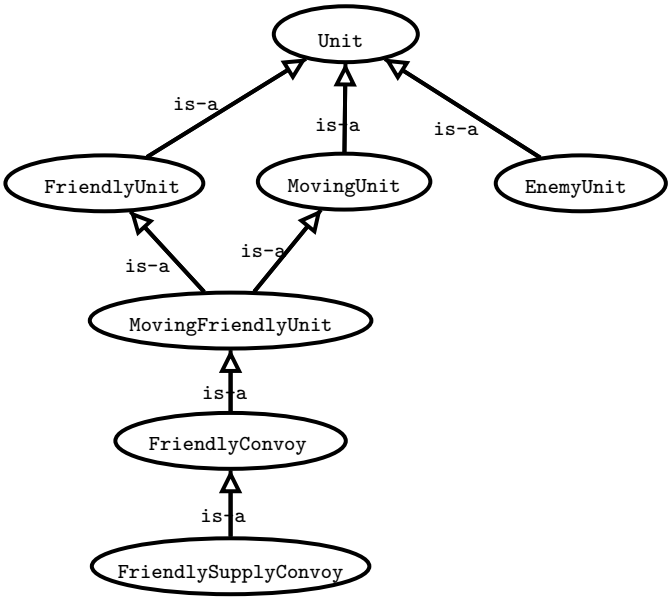


Figure 4.1: A fraction of a unit taxonomy.

hold, so a subscription for events involving `FriendlySupplyConvoy`s should not match a notification about the more general `FriendlyConvoy`.

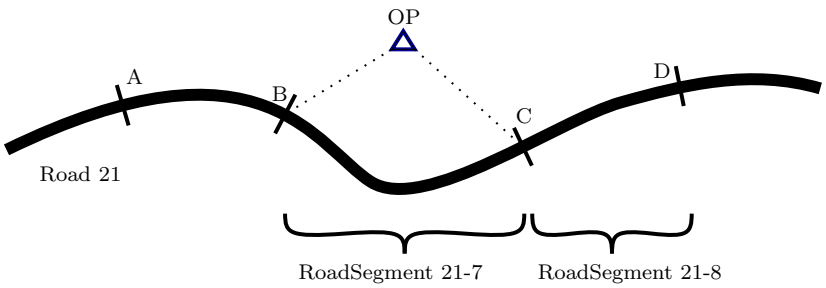


Figure 4.2: A sketch of road 21. An observation post (OP) watches segment 21-7 between ponts B and C.

More involved reasoning might also be possible. For instance, the OP from section 3.2.3 might subscribe to notifications about anything approaching the particular stretch of a road that it observes; `roadSegment21-7` between points B and C (figure 4.2). A convoy travelling along the same road would possibly not be aware of the OP or the points B or C. To keep the world informed, though, it would publish a notification when passing point A, declaring its intention to continue to point D. With information about the sectioning of the road and an applicable set of rules, a semantic broker should be able to figure out that the convoy is indeed approaching `roadsegment21-7`,

and that the notification matches the subscription.

4.2 Event representation

In a notification service that uses the contents of event notifications for matching and routing, some sort of structured text is most easily processed — as opposed to natural language, sound or images. This is where semantic technologies enter the scene. An RDF graph can represent practically any information, whilst at the same time being semi-structured. Strictly speaking, RDF is not a textual format, but a *data model* with object–predicate–subject triples as building blocks. There are, however, standards for serialisation of RDF graphs as text, most notably RDF/XML[5].Once expressed in a standard text format, the graph can be exchanged across networks in an application-agnostic form. Alternatively, RDF-graphs in application-specific formats, such as Java objects, can be exchanged between more tightly coupled components. In a loosely coupled pub/sub system, a text representation seems to be the most suited.

An event in the real world can be represented in a variety of formats in supporting information systems. RDF can be used to describe any event, either as the primary representation or as metadata about images, video or sound that make up the primary — and presumably richer — representation. When the RDF graph itself provides the primary description of an event, the content of the notification will be the serialised graph and nothing else. When the graph describes some other representation, there are two possibilities: The first is to include the primary representation in the notification along with the RDF representation. The other possibility is to make the resource retrievable across the network by providing a URI or some callback-mechanism. The former is acceptable when the primary representation is relatively small (measured in Bytes), for example an unstructured text document. In that case, the primary purpose of the RDF-graph would be to annotate the document so as to make it machine processable. The latter would be the better choice if more expensive multi-media formats are involved — at least as long as some proportion of the recipients can be expected not to retrieve the multi-media file. That way, valuable bandwidth will be saved. RDF can be used for both data and metadata. The distinction between what is data and what is metadata is not always clear-cut, but that is of minor significance to the notification service: An RDF representation will form the notification content in any case.

Figure 4.3 shows a small example of a `MovementEvent`. It includes potentially useful information such as information needed to get in contact with the convoy one-to-one should the need arise. The same graph is serialised in Notation3[3] below.

```
@prefix :        <http://example.org/events/someMovementEvent#> .
```

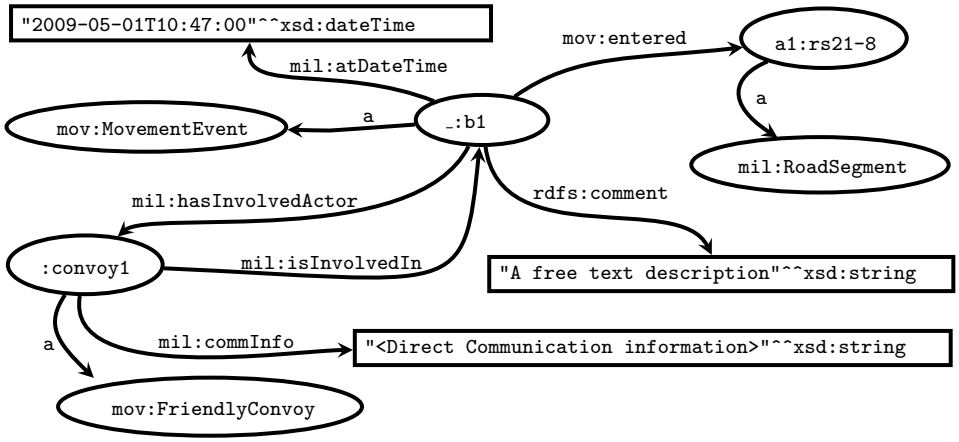


Figure 4.3: An RDF graph of a simple `MovementEvent`.

```
@prefix a1:      <http://example.org/baseAssertions/someAreaInfo#> .
@prefix mil:      <http://example.org/ontologies/milOntology#> .
@prefix mov:      <http://example.org/ontologies/movementOntology#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .

:rs21-8
    a       mil:RoadSegment .
_:b1
    a       mov:MovementEvent ;
    mil:hasInvolvedActor :convoy1 ;
    mov:entered :rs21-8 ;
    mil:atDateTime "2009-05-01T10:47:00"^^xsd:dateTime ;
    rdfs:comment "A free text description..."^^xsd:string .
:convoy1
    a       mov:FriendlyConvoy ;
    mil:commInfo "<Direct communication information>"^^xsd:string ;
    mil:isInvolvedIn _:b1 .
```

For the sake of consistency in event representation, it would be useful to define a basic model of events in an ontology. The ontology should be fairly general, and contain only classes and properties that are common to all kinds of events, expressing such as who, where, when etc. This model could be part of a general military domain ontology, or it could even be in a separate *event-ontology*. In addition, specific activities and application types will have specific needs that could be expressed in more specialised ontologies, extending the general one. For the convoy scenario, a *movement-ontology* could define a `MovementEvent` as a subclass of `Event` from the general ontology. And an *intelligence-ontology* could define an

`EnemyObservationEvent`. Every concrete event graph would then include an instance of the appropriate `Event` subclass and what else might be defined in the model. Other than that, the full flexibility of RDF is available for the construction of event graphs.

The makeup of the set of ontologies is a separate issue. A few comprehensive ontologies would favour consistency, whereas a bigger number of small, specially tailored ontologies can make reasoning a bit less expensive. That said: the actual contents, breakdown and interrelations of ontologies is primarily an engineering and modelling task — a big one, and out of scope of this discussion about the principles of a notification service.

The mechanics behind the actual construction of RDF representations of events is not a concern for the notification service, but left to the client applications using the service. Presumably, that would some times be a fully automatic process, and some times a partly manual process. Even though someone familiar with the RDF format can make sense of them, the various RDF syntaxes are not meant for human readers or writers. Therefore, some kind of user-friendly interface should allow the user to express his needs, and then convert that to RDF. Ideally, client applications would themselves be semantic-aware, and use RDF as part of their internal data representation. In that case, exchanging RDF graphs would be a very direct form of interoperation. Otherwise, some conversion will have to take place between the native format and RDF as the exchange format. Either way, the notification service will make use of the RDF graph with the prime purpose of filtering notification by matching them with registered subscriptions.

4.3 Subscription representation

In principle, subscriptions can be expressed in any format as long as the broker has a way to determine if a notification matches a subscription or not. With the event notification represented as an RDF graph, there is more than one possible way to express subscriptions and do the matching with the event graph.

One option is for the subscriber to declare its own subscription ontology which specifies the event of interest as a defined class; for example `MyInterestingMovementEvent`. This allows a standard DL reasoner to check wether or not the published event is an instance of the defined class — in which case it is deemed to match the subscription. This approach is adopted by Li and Jiang in their *Semantic Message Oriented Middleware*[50], in the document retrieval service of Haarslev and Möller[35], and in the *Semantic Infosphere* described in [77].

Another way is to construct a pattern of triples that resembles an RDF graph. The pattern defines the shape of the graph and also constraints on some of the edges and nodes. Matching becomes a question of identifying

subgraph isomorphism between the subscription graph pattern and the event graph. An event graph of the same shape that satisfies the constraints in the graph pattern is found to match. This approach is Used in the *Ontology-Based Publish/Subscribe System* (OPS)[80] and in the G-ToPSS system[60].

The third option, and the one explored in most detail here, is to describe the requested information with a suitable query language. A number of languages to query RDF have been proposed, and they differ both in syntax and expressivity[36, 44, 21]. To stay with the W3C stack of technologies, the W3C recommendation SPARQL[6] will be the language of choice here.

A SPARQL query defines one or more graph patterns with certain constraints that the queried RDF graph needs to satisfy. Syntactically, the graph patterns are expressed in the where-clause of the query, and it allows for both optional and alternative sub-patterns. Also, a single query can be executed across several distinct datasets. Without going into a detailed presentation of SPARQL, these and other features makes it possible to define relatively sophisticated queries. With the subscriptions represented in SPARQL and the events represented in RDF, any compliant query engine can do the actual matching by running the queries against the event graphs.

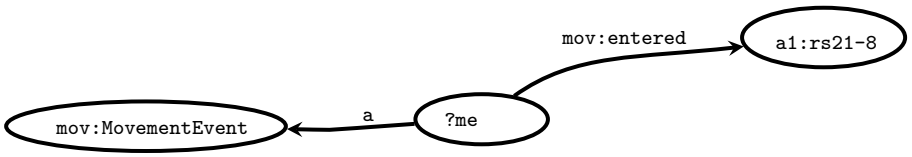


Figure 4.4: A simple subscription graph pattern with variable `me`.

A simple graph pattern for any notification about a `MovementEvent` expressing that someone has entered the specific `RoadSegment` with URI `<http://example.org/baseAssertions/someAreaInfo#rs21-8>` is presented graphically in figure 4.4. The same graph pattern is captured in the SPARQL query below. This query will match the event in figure 4.3.

```
prefix mov:      <http://example.org/ontologies/movementOntology#> .
prefix a1:      <http://example.org/baseAssertions/someAreaInfo#> .

ASK
WHERE
{ ?me a mov:MovementEvent ;
     mov:entered a1:rs21-8 .
}
```

In this case, the query matches the event graph directly, without support from any reasoning. This is useful enough, but hardly representative of the powers of semantic technologies. The next section presents an example that goes one step further and makes use of simple inference and rule processing to have the matching succeed.

4.4 Example case

To provide examples of 'proper' semantic matching, figure 4.5 shows five imaginary nodes connected to the notification service (NS): The two upper nodes are convoy1 (C1) and convoy2 (C2), both publishing routine notifications of progress. The lower three nodes are the brigade logistics staff (G4), the observation post from section 3.2.3 (OP) and the headquarters of all Norwegian forces in the theatre of operation (HQ NO). Each has issued one subscription within its sphere of interest. Informally, the respective subscriptions are for notifications about:

G4: Any supply convoy belonging to brigade 3.

OP: Anything moving toward the observed section of road 21.

HQ NO: Any Norwegian unit moving around.

Excerpts from subscription queries and notifications graphs in Notation3-like syntax are shown in the figure. Figure 4.2 on page 37 shows a sketch of the imaginary road.

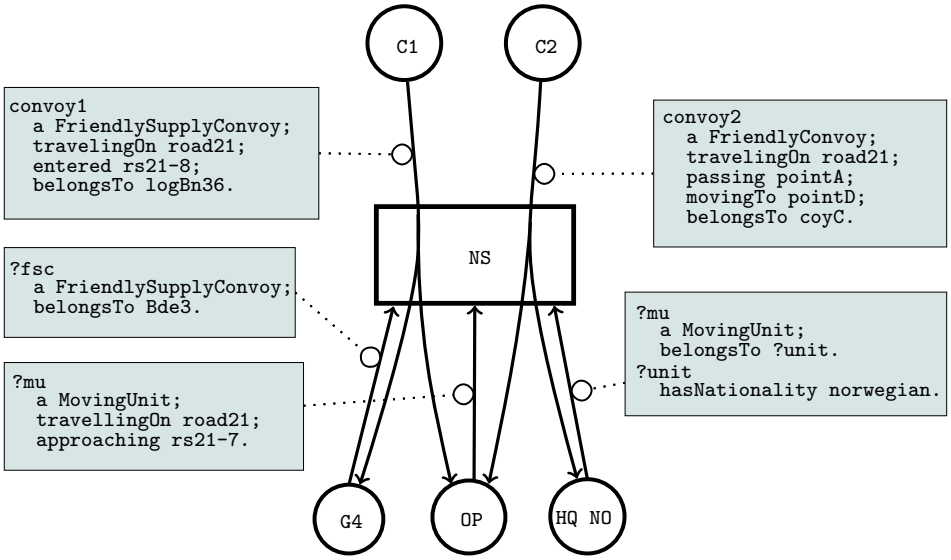


Figure 4.5: Notifications from the producers (C1 and C2) are filtered and routed to the right subset of consumers (G4, OP and HQ NO) based on the semantics of the contents.

With syntactic matching, none of the subscription would be satisfied as given. To show how semantic matching can improve on this, the listing below briefly comments what extra information needs to be available, and what

inferences must be made for the match to occur. Alternatively; when there is no match, the reason is explained. The unit taxonomy from figure 4.1 is still valid. Namespace prefixes are omitted for brevity.

G4–C1: The triple `(logBn36 belongsTo Bde3)` must be asserted, and `belongsTo` must be an `owl:transitiveProperty`. Then, `(convoy1 belongsTo Bde3)` can be inferred as well, and the query will match.

G4–C2: `FriendlyConvoy` is a more general concept than `FriendlySupplyConvoy`. Thus, `convoy2` cannot be classified as a `FriendlySupplyConvoy`.

OP–C1: Something like the triple `(rs21-7 nextTo rs21-8)` must be asserted. Then, a rule like the following would entail `(convoy1 approaching rs21-7)`:

```
entering(?obj, ?roadSegmx) ∧
nextTo (?roadSegmx, ?roadSegmy)
→ approaching(?obj, ?roadSegmy)
```

The problem with this solution is that the same kind of assertion would be inferred every time something crosses from `rs21-7` into `rs21-8` as well — in other words when leaving the observation area. This is hardly what the observation post wants. With the information given, this is hard to get around. One possible solution would be to construct a more comprehensive model which specifies directions and orders the road segments.

OP–C2: This case is similar to the previous. To build on the idea of directions and orderings, the triples `(pointA westOf rs21-7)` and `(rs21-7 westOf pointD)` should provide a good starting point. Next step is to add the rule:

```
westOf (?pointx, ?area) ∧
westOf (?area, ?pointy) ∧
passing (?obj, ?pointx) ∧
movingTo (?obj, ?pointy)
→ approaching (?obj, ?area)
```

From this, `(convoy2 approaching rs21-7)` could be found.

Depending on what other assertions are available, the above rule might produce more assertions about approaching objects than strictly called for. To keep the number of very remote 'approaching objects' down, it

might be prudent to add a couple of clauses to the rule: One reasonable restriction is that the points and the area all be positioned along the same axis of transportation (on or off road). Another could set an upper limit for the distance between the point passed and the area. If not, some very early warnings might be given.

HQ NO–C1: As an instance of `FriendlySupplyConvoy`, `convoy1` can also be classified as a `MovingUnit`. So when this query fails to match, it must be because the triple `(logBn36 hasNationality norwegian)` is not to be found.

HQ NO–C2: Like the previous case, `convoy2` is also a `MovingUnit`. Since the notification matches the subscription, `coyC` is apparantly Norwegian.

These examples are mere illustrations of matching on semantics. The information model they implicitly build on is too simle to to be useful for practical purposes. None the less, they serve as a demonstration of how general purpose ontologies, rules and base assertions can be used in a notification service with matching capabilities that go well beyond the limitations of syntactic matching.

After this introduction to what semantic filtering and routing is about, it is time to move on to an outline of what the apparatus needed might look like. The next section will describe the high-level architecture of a single broker in the semantic notification service, and how it handles subscription queries and event notifications.

4.5 Routing mechanics

The network of brokers collaborate to bring notifications from producers to consumers as explained in section 3.1.6. To bring a notification from one client to another, the notification is passed in hops from one broker to the next until it ends up with the consumer. Each broker works as a filter, letting through only the notification it has an active subscription for. It also works as a router, selectively forwarding the notification towards the final consumer. This section will outline a possible internal architecture of a broker and the way it performs basic filtering and routing.

4.5.1 Broker components

The internals of a broker would look something like figure 4.6. This illustration can be seen as an exploded view of one of the squares in figures 3.4 on page 28.

First, there is an *Input queue* that accepts incoming messages and acts as a buffer. The broker is indifferent to what kind of node it receives the

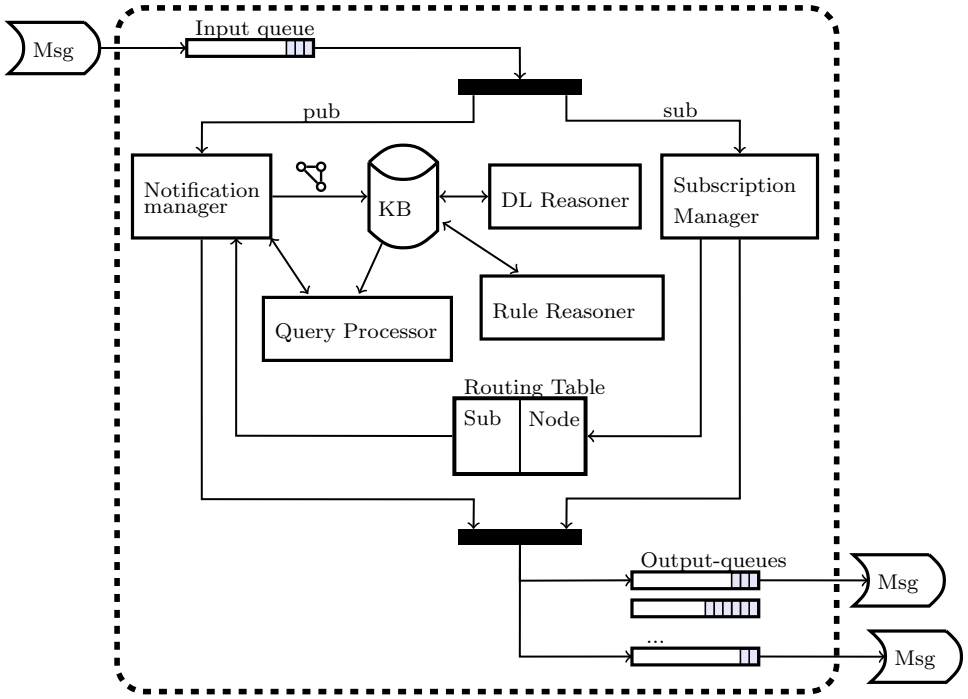


Figure 4.6: Components of a single broker in the notification service (within the dotted border). Messages are received in the upper left corner and sent from the lower right.

message from. Messages from local clients and neighbouring brokers enter the same queue — possibly ordered by priority as indicated in the message head.

Messages contain either published notifications (*pub*) or subscriptions (*sub*). There are two separate managers responsible for handling the processing of each of the two varieties: The *notification manager* and the *subscription manager*.

A *knowledge base (KB)* holds ontologies and a background dataset that will be used for reasoning over the event graphs. The ontologies would be all the agreed-upon ontologies that can be referenced in the incoming events. Because of the limited network resources, events graphs themselves should be kept small. They should only describe the event itself, making reference to the common ontologies for classification and further inference. To provide the necessary context for an event, a relatively stable set of instance data, or base assertions, needs to reside in the knowledge base of each broker. That would for instance include geographical information such as the relative location of points A,B,C and D from figure 4.2, or technical data about vehicles. In *description logic (DL)* terminology, the ontologies would make

up the *terminology box* (TBox) of the knowledge base, and the base assertions together with the inserted event graph would be the *assertion box* (ABox). A sufficiently rich set of base assertions will make it possible to infer additional useful facts about events concisely described in the notification. A standard *DL reasoner* will be used to do the reasoning over the knowledge base.

Due to limitations in the expressiveness of OWL, certain kinds of useful inferences cannot be made from ontological constructs alone[64]. An oft-cited example is the `hasUncle` property that can be composed of the `hasParent` and `hasBrother` properties[43]. Unfortunately, OWL has no way to express composition of properties. A rule language, such as the Semantic Web Rule Language (SWRL)[43], can help overcome some of the limitations. Thus, a set of rules complementary to the ontologies might be included in the knowledge base. A compliant *rule reasoner* processes the rules and adds new assertions back to the knowledge base.

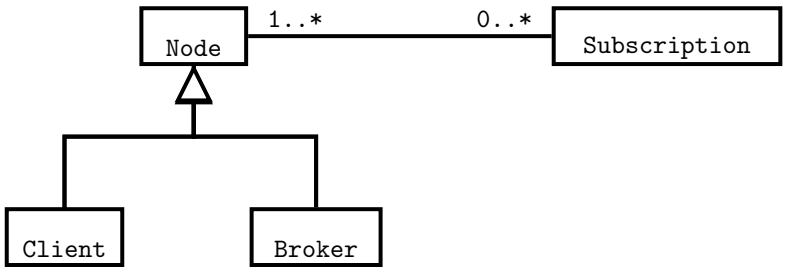


Figure 4.7: A UML class diagram of the association to `Subscription`.

Then there is the *routing table*, which is the central datastructure in any pub/sub broker. The routing table maps subscriptions to nodes. In principle, a broker interacts the same way with directly connected clients as with neighbour brokers. Therefore, they are both represented as nodes in the routing table. Each node may register multiple subscriptions, and more than one node can register the same subscription (figure 4.7). In a naive design, the routing table is simply a flat table where subscriptions are either identical or not. More refined schemes can take advantage of similarities and partial overlaps between subscriptions for more efficient filtering. We'll briefly come back to that in section 5.2.3.

With the subscriptions expressed as SPARQL queries, a *query processor* must be included to run the queries against the event graph enriched with assertions from the knowledge base.

Finally, the *output queues* hold the messages destined for neighbouring brokers and subscribers — also possibly ordered by priority. Because the associated nodes cannot be trusted to be connected at all times, each one needs to be assigned a unique output queue. The queues provide the store-and-forward capability of the notification service, allowing subscribers to collect messages that came in while the subscribers have been off line. Also,

the queues make it possible to bundle several small messages into one. That way, the total load on the network can be reduced.

4.5.2 Routing process

Using queries to express subscriptions to events can be seen as the inverse of submitting a query to a database: Instead of storing data, the assembly of queries is kept in the notification service. And instead of entering a query at a time and evaluating it against the stored data; the event data are entered in discrete chunks, each evaluated one at a time against the stored queries.

Events are processed one at a time in the broker. A message has a head for transmission purposes, and a body containing the payload; an event graph or a subscription query. Incoming messages are buffered in the input queue. As soon as processing capacity becomes available, the next message is dequeued and entered into the appropriate manager.

When a message enters the notification manager, the event graph is extracted from the message. In principle, this bare event could be evaluated as is against the subscription queries. After all, it is a representation of an `Event` instance with a set RDF triples to describe it, and that graph can be queried. However, that would not add much compared to a set of key–value pairs found in standard content-based filtering. The reason is that SPARQL queries do not go beyond the information explicitly expressed in the graph they are run against. So if, for example, a `MovementEvent` about a `FriendlySupplyConvoy` is entered, it would not match subscriptions for `MovementEvent`s about `FriendlyConvoy`s, even though it should from the semantic meaning of it: Any instance of `FriendlySupplyConvoy` is also an instance of `FriendlyConvoy`, but the query engine does not know. Therefore, the fact that this particular individual is an instance of `FriendlyConvoy` has to be added to the graph first. The bare event graph is added to the knowledge base and merged with the base assertions. Then, the reasoners are applied to find all possible entailments from the ontologies and rules.

With all entailments disclosed, its time to check if the event matches any of the registered subscriptions. The notification manager iterates through all subscriptions in the routing table. One subscriptions query at a time is passed to the query processor, which runs the query against the assertions in the knowledge base. The result of the query is returned to the notification manager. If the query is found to match, the notification manager knows from the routing table which node(s) issued that particular subscription, and the message is put in the corresponding output-queue(s). In the basic design, the incoming message will be passed on just as it arrived, without adding any of the inferred entailments.

If no match is found, the message is not put in any of the output queues, and consequently travels no further. In practice, that will only happen for messages coming directly from a connected client. If a message comes

in from a neighbour broker, it is because the message is backtracking a subscription trace, as shown in figure 3.5 on page 28. Clients, on the other hand, publish event notifications without knowing whether or not there are any subscriptions for it. This is how notifications gets filtered and possibly dropped.

Four query forms are defined in SPARQL: SELECT, ASK, CONSTRUCT and DESCRIBE. Of these four, the simple ASK-queries satisfy the basic requirement; to decide whether or not the constraints in the graph pattern of the subscription are satisfied by the event graph. The query returns a boolean, and if it evaluates to `true`, a match is found. The query does not need to describe all of the event graph to match. It should only express the interest of the subscriber, and then the rest of the graph comes along.

After all the subscriptions have been run through, the knowledge base is reset. That means removing the event graph under scrutiny, as well as the entailments infered from it. This is necessary to avoid false hits when subscription queries are run against the next event graph. With old events lingering in the knowledge base, the subscription queries that match them will return `true` no matter what new event is entered.

Notifications coming from clients and neighbour brokers are in principle treated the same way. After all, the broker is indifferent to what happens to the notification after it has been forwarded — if it reaches its final destination or is passed on further. The only responsibility the broker has is to decide where the message is going next, and send it there. Connections between brokers may be of a different quality than connections to clients, but the filtering and routing process will be the same — one hop at a time.

So far, this description has been about handling notifications. The other path through the broker is taken whenever the message contains updates to the registered subscriptions. There is nothing 'semantic' about the administration of subscriptions, so I will not discuss this issue in great detail here. None the less, a few words of explanation are called for.

Three main types of updates occur: Enter a new subscription; subscribe(sub), delete an existing; unsubscribe(sub), or prolong the validity of an existing; renew(sub). The third type is necessary because subscriptions need to have an expiry time. Otherwise, obsolete subscriptions would pile up and cause unnecessary load on both the brokers and the network. As noted in section 3.1.2, subscriptions are wrapped in messages and transmitted between brokes the same way notifications are. The chief difference between dissemination of notifications and dissemination of subscriptions, is that any new subscription is forwarded to all neighbour brokers without filtering — except the broker it came from. Also, subscriptions are forwarded only to brokers, and not to clients.

in summary, a brief and informal algorithm for the routing process in an individual broker looks as follows:

```
Dequeue message from the input-queue.
```

```
If message contains notification:
    Get event graph from the message.
    Add event graph to knowledge base.
    Run reasoners on the merged knowledge base.
    For each subscription in the routing table:
        Run subscription query against the knowledge base.
        If they match:
            Add message to output-queue of the appropriate node(s).
If message contains subscription:
    Make adjustments to routing table.
    Add message to output queue of remaining neighbour broker(s).
```

For the broker network as a whole, there is no overarching control structure or algorithm — no grand plan. The activities of the notification service is equal to the collective activities of its constituent parts; the brokers. The notification routing is truly decentralised.

4.6 Composite events

A very powerful and exciting possibility of using semantic technologies in publish/subscribe systems, is the capacity to detect composite events. A *composite event* is defined in [48, p 253] as "a pattern of event occurrences of interest to a subscriber. These patterns may express temporal or causal relationships between different events". In other words, an event that is not explicitly represented in any one notification may be implicitly contained in events described in two or more unrelated notifications. Evidently, these *primitive events* may themselves be interesting to subscribers, but as far as the composition is concerned, it is the relations between them that matter. Inspecting the relations can reveal the implicit event that in a sense is of a higher order than the primitive events. A composite event may also be viewed as an *abstraction* of the primitive events, indicating that it expresses a more general concept than the sum of the details. In turn, composite events can themselves be further composed into even higher order composite events in a recursive manner.

4.6.1 Benefits of composite events

The primary effect of event composition is, of course, to maximise the information that comes out of the data. Primitive events are valuable carriers of information in their own right. And they become even more valuable if even more information can be derived from combining them and revealing patterns, trends, or even inconsistencies in the data. Seeing them in context can bring out a bigger piece of the overall picture. Events that seem insignificant in isolation can prove to be valuable when seen together.

Another very welcome effect, particularly in disadvantaged grids, is that one composite event notification can replace several lower-level ones, saving

both network traffic and routing processing. Not to mention that it saves processing in the subscribing nodes to aggregate and make sense of the details — processing that even will be unnecessarily duplicated if more than one information consumer is actually after the same high-level information. If only the composite event is interesting to anyone, the sooner the primitive events can be discarded, the better. This way of aggregating data is similar to the information fusion that takes place in wireless sensor networks[54], both in motivation and in general approach.

4.6.2 Event composition with semantic technologies

The related events have to be discovered in some way, and the relation of interest has to be described. One approach is to express subscriptions for composite events in a special *composite event language*[53], or *composite subscription language*[48] that allows the subscriber to define the composite event in terms of the underlying primitive events. With this method, a subscription is not for one single high-level event, but for a logical combination of elementary events[27]. One thing these approaches have in common is that they build on event matching mechanisms that are syntactic rather than semantic. So even if the composition mechanism is effective, it cannot escape the inherent limitations of syntactic matching.

The alternative approach offered here, is to use semantic technologies to perform the event composition. Semantic technologies lend themselves well to this sort of event detection. Essentially, inferring composite events is the same as inferring any kind of abstractions from more primitive pieces of information. As it happens, that is exactly one of the things semantic technologies are designed to do. With the events represented in RDF, all the reasoning capabilities of semantic technologies are available, including rule-based reasoning and inferencing over ontologies that define the semantics of the events.

4.6.3 Example case

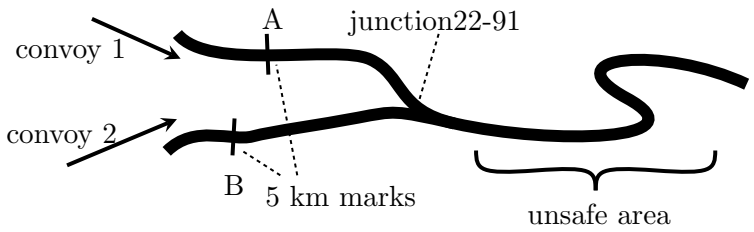


Figure 4.8: Two convoys approaching a stretch of road that runs through an unsafe area.

As an example use case, consider a situation where two convoys plan to drive along the same stretch of road through an unsafe area (figure 4.8). With a certain separation between the two convoys, the risk is acceptable. But if they enter the road at more or less the same time, they might run into trouble. Either they will have to cram the vehicles together, leaving them with less space to maneuver in the case of an attack, or one of the convoys will have to stop and wait for a while. Both options will leave them more vulnerable, and should be avoided if possible. To predict this 'traffic jam in the making', both of them could publish a notification of the event that they cross an imaginary line less than, say, five kilometers from the entry to the critical stretch of road. It would be reasonable to assume that sort of running progress reports as a matter of routine. If both convoys pass the 5 km mark with less than a five minute interval, the distance between them might be smaller than it ought to be, and a notification about a possible congestion should be constructed. This warning can be interesting to actors that would not otherwise care about the exact positions of both convoys. The convoys themselves are one example. Convoy commanders should not be left to keep track of all other moving units 'by hand'. Instead, they should be alerted when a situation needs to be handled, such as a possible congestion. Similarly, the home units of the convoys would probably monitor their own convoy in detail, but only maintain a high-level picture of all other activities in the greater theatre of operation. Once something threatens to interfere with their own activities, however, they need to know.

The fact that the congestion-event has not occurred yet goes to show that a composite event can refer to events and relations that are not directly observable. Projected and predicted events like that are in a sense less real than events that already happened in the physical domain, but that does not make them any less important to decision makers.

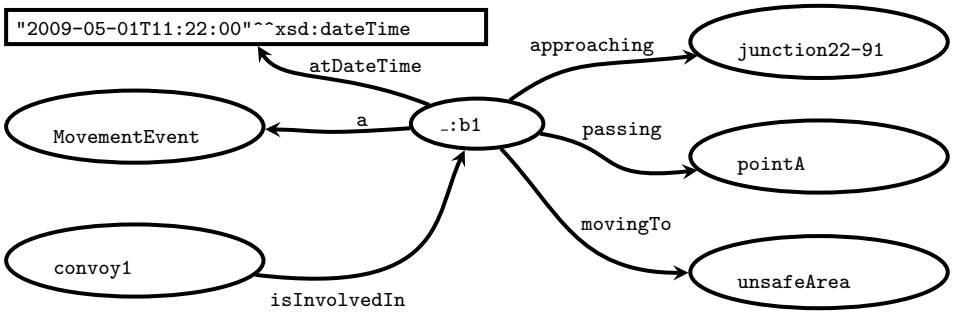


Figure 4.9: An instance of `MovementEvent`.

The two convoys in question, `convoy1` and `convoy2`, would issue notifications as shown in figure 4.9. The individual of class `MovementEvent` is a

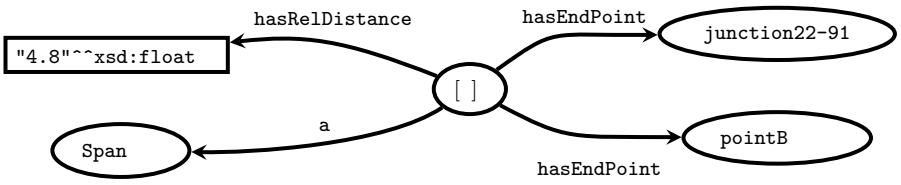


Figure 4.10: A span between two points with their relative distance specified.

blank node with properties describing it. Classes and namespace prefixes are omitted for brevity. With some additional base assertions, a SWRL rule can be used to check if any two out of a possibly large number of `movementEvent`s present a possible congestion. If so, the rule will create a `CongestionWarning`, which is a subclass of `MovementEvent`, alerting interested clients about the situation. Next, I will present such a rule, and as it is rather involved, I will present it in several small chunks:

- The first question to be answered is: Do two `movementEvent`s exist which indicate that two convoys are moving towards the same junction and then on to the same area?

```
MovementEvent(?movEv1) ∧
MovementEvent(?movEv2) ∧
FriendlyConvoy(?conv1) ∧
FriendlyConvoy(?conv2) ∧
isInvolvedIn(?conv1, ?movEv1) ∧
isInvolvedIn(?conv2, ?movEv2) ∧
passing(?movEv1, ?pt1) ∧
passing(?movEv2, ?pt2) ∧
approaching(?movEv1, ?junc) ∧
approaching(?movEv2, ?junc) ∧
movingTo(?movEv1, ?area) ∧
movingTo(?movEv2, ?area) ∧
...
```

- Additionally, the two variables `conv1` and `conv2` should refer to different units:

```
...
abox:hasURI(?conv1, ?uri1) ∧
abox:hasURI(?conv2, ?uri2) ∧
swrlb:notEqual(?uri1, ?uri2)
...
```

Two built-ins are used for this; `abox:hasURI` and `swrlb:notEqual`. The latter belongs to the core SWRL built-Ins defined by the SWRL submission[43]. The former is part of the built-in libraries defined and implemented in the ontology editor Protégé[4, 7]. Technically, this could be more directly expressed with `owl:differentFrom`, but different units publishing two `movementEvent`s will generally not be aware of one another, and therefore never assert that they are not the same. It can, however, be trusted that they are referenced with different URIs.

- The second question is: Are the points passed five kilometers or closer to the junction where the moving units will meet?

```
...
Span(?span1) ∧
Span(?span2) ∧
hasEndPoint(?span1, ?pt1) ∧
hasEndPoint(?span2, ?pt2) ∧
hasEndPoint(?span1, ?junc) ∧
hasEndPoint(?span2, ?junc) ∧
hasRelDistance(?span1, ?dist1) ∧
hasRelDistance(?span2, ?dist2) ∧
swrlb:lessThanOrEqual(?dist1, 5.0) ∧
swrlb:lessThanOrEqual(?dist2, 5.0) ∧
...
```

  The answer to this is not to be found in the event graphs alone. In addition, the broker needs to have some background assertions in the knowledge base to determine this. Here, that would be simple RDF graphs with a blank node of type `Span` expressing the distance between two points. One such is shown in figure 4.10.

- The third question is: Did the `movementEvent`s occur within a five minute interval?

```
...
atDateTime(?movEv1, ?timeStamp1) ∧
atDateTime(?movEv2, ?timeStamp2) ∧
temporal:notBefore(?timeStamp1, ?timeStamp2) ∧
temporal:duration(?interval, ?timeStamp1, ?timeStamp2,"seconds")∧
swrlb:lessThan(?interval, 300) ∧
...
```

Like the builtin `abox:hasURI` above, the temporal-builtins are also part of the SWRLTab libraries[7].

- For any solution satisfying all of the above — that is, whenever two `movementEvent`s are found to indicate that two convoys approach the same road at about the same time — a `congestionWarning` is constructed:

```
...
swrlx:makeOWLThing(?newInd, ?interval)
→ CongestionWarning(?newInd) ∧
hasTimeGap(?newInd, ?interval) ∧
isComposedOf(?newInd, ?movEv1) ∧
isComposedOf(?newInd, ?movEv2) ∧
isInvolvedIn(?conv1, ?newInd) ∧
isInvolvedIn(?conv2, ?newInd) ∧
atDateTime(?newInd, ?timeStamp1)
```

Another Protégé built-in, `swrlx:makeOWLThing`, is used to create a new individual (figure 4.11). The new individual gets classified as a `CongestionWarning` and given descriptive properties, such as the most recent timestamp of the two primitive events. Furthermore, it gets linked to the primitive events and the involved convoys with the properties `isComposedOf` and `isInvolvedIn`.

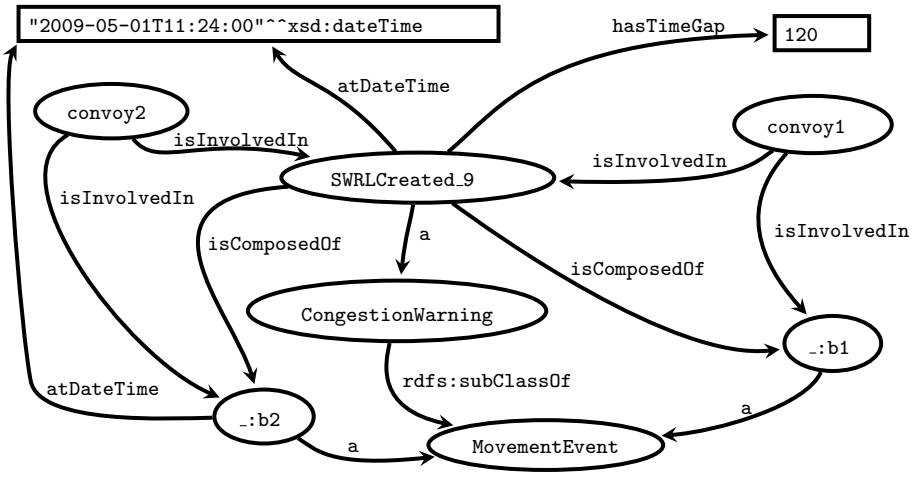


Figure 4.11: An instance of `CongestionWarning` (`SWRLCreated_9`) composed from the graph in figure 4.9 (`_:b1`), another event of the same kind (`_:b2`) and the geographical base assertion expressed in figure 4.10.

`CongestionWarning` is a subclass of `MovementEvent`. Hence, subscription queries for events with the right set of properties will match it. If, for example, the home unit of one of the convoys subscribes to all events the convoy `isInvolvedIn`, the notification service should present the new `CongestionWarning`.

4.6.4 Detection of inconsistencies

One possible use case for event composition is discovering inconsistent notifications. With careful modelling, semantic technologies can be used to advantage for detecting inconsistencies in the instance data. For example, if a convoy publishes a notification reporting that it passed some `pointA`, and another notification appears five minutes later, stating that the same convoy passed `pointB` 60 km from `pointA`, something is clearly wrong. Military convoys do not travel with the speed of 720 km/h. Either the points are mislocated, the identities of two different convoys are mixed up, or the type of the moving object should have been `Aircraft`, just to mention a few possibilities.

Once detected, it is necessary to bring the inconsistency to the attention of a human decision maker. Realistically, only a human can clarify and rectify the situation. The natural way to do that is by issuing a notification that describes the case. Since this is a matter of relating two or more different notifications, it makes sense to handle it as a case of event composition.

The inconsistencies referred to above are with respect to the real world. Internal inconsistencies with respect to the information model, or the OWL semantics, is another matter. For example, no individual can be an instance of two disjoint classes. A DL reasoner can detect the inconsistency, but it cannot use it for constructing a composite event.

4.6.5 Event composition in the broker

Events come together in the brokers, so the broker stands out as the right place to construct composite events. A simple matching mechanism as described in section **??** will not discover these implicit relations because it evaluates each notification in isolation: The broker receives a notification, checks it against the currently registered subscriptions to see if any of them match, passes it on if it does, and then discards it. When the next notification is processed, the broker has no recollection of the previous ones. The pub/sub interaction pattern is stateless in its basic form.

Keeping a record of past events is evidently necessary to detect composite events: The connection between events P and Q only appears if both P and Q can be seen together. With such a record in place, there must be some way to reveal the connection.

The architecture of the broker has to be modified somewhat to accommodate event composition: Received event graphs need to be collected in a triple store. As noted in section 4.5.2, historic event graphs need to be kept away from the current one when the subscription queries are run. Each event graph should be kept for as long as it is likely to contribute to new entailments. After that, it should be removed to keep the data volume down and thereby ease the inferencing process. In addition, there needs to be a proper combination of ontologies, rules and queries at hand. These may collectively be called the *compositon definitions*.

The composition queries will be different from the subscription queries that are run aginst individual event graphs. The composition queries will be of the DESCRIBE or COMPOSE form, returning graphs upon match. They will be applied repeatedly to the repository of past events as it fills up with incoming events. That way, they form new event graphs that can be added to the primitive ones and in turn be queried the same way. To escape the problem of reappering entailments, the inferred graph from one iteration will be added to the base assertions before the next iteration. That way, only newly discovered assertions will appear in the inferred graph of the next iteration.

Fitted with a set of composition definitions, reasoners should have what they need to detect the composite events. The matching process will proceed in two stages: First, any composite events are found Then, the subscription queries are run over the union of entailed composite events (if any) and the enriched event graph as in the original design.

Now, one questions remains: Where should the compositon definitions be formed? Two options present themselves: The subscribers make their own definitions and issue them the same way as ordinary subscriptions, or a set of common definitions reside in the brokers, together with the common ontologies and rules.

The first option follows the lines of the customised event composition languages of [53, 27] — only in this case, standard semantic technologies are employed. Small ontologies, rule sets and queries may all make part of the body of subscriptions. In the broker, the ontologies and rules will be included in the knowledge base TBox, where they will contribute to the discovery of new triples. Composition definitions coming from the clients will be managed like other subscriptions.

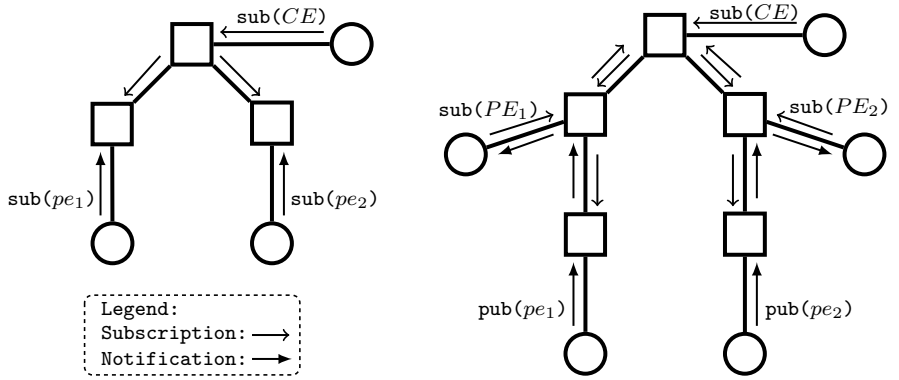
The second option — keeping predefined definitions in the broker — is to a large extent supported by the broker design as is. The knowledge base TBox is there, with ontologies and rules to support inferencing. All that needs to be added is the right constructs in the ontologies and rules and a set of composition queries.

The two options are not mutually exclusive, and both will need to be in place. The choice of which to choose for a specific composition definition should be based on their pros and cons. Composition definitions that are

predefined and packed with the broker, so to speak, are more static than the ones defined by clients and issued with subscriptions. On the other hand, handling subscriptions consumes network resources, as well as some resources in the brokers. Because of this, predefinition of composition definitions will be preferable for predictable and stable cases. Less common and more short-term compositions had better be defined by the subscriber itself. But this, again, is an engineering and modelling decision from case to case.

4.7 Elicitation of composite events in the notification service

Event composition should work well as long as the primitive events are collected in the same broker. In the trivial case of one centralised broker for the whole notification service, all events come together, and the full possible set of entailments can be inferred. With a network of brokers, however, matters become more complicated.



(a) No subscriptions for primitive events.

(b) Notifications of primitive events are routed through different parts of the network.

Figure 4.12: Composition of events fails if the primitive events do not come together in any broker.

Each broker does its reasoning in isolation from the others. So for a composite event to be disclosed, the primitive events that imply the composite event must come together in the same broker. When no subscription is registered for a particular kind of event, however, any published notification about such an event is dropped by the first broker receiving it from the publisher. So if no one subscribed to the primitive events, and they are published to different brokers, they will never come together (figure 4.12a). Thus, the composite event will never be inferred. Clients can generally be expected to subscribe to high-level composite events instead of the primitive

events they are derived from — and rightly so. However, if only composite events are in demand, they won't become available because the primitive events will be dropped too soon.

Also, consider a situation where subscriptions to the primitive events are in fact registered, but in different parts of the broker network (figure 4.12b). The messages will be routed towards the subscribers, but the paths they follow may never intersect. Again, the composite event will remain undiscovered. In figure 4.12, the primitive events pe_1 and pe_2 would have led to the composition of ce if they had been brought together. The composite event ce would in turn have matched the subscription for CE, but the way the primitive events are propagated, that never happens.

To get around this situation, the notification service should have a mechanism that increases the probaility of discovery of interesting composite events that are potentially available from the primitive events. One solution might be to have the clients subscribe to the primitive events instead of — or perhaps in addition to — the composite events. That would, however, run counter to the goal of reducing network traffic. Moreover, because the routing tables would grow, the cost of matching and routing would increase. Another solution might be to flood all notifications through the entire notification service, making sure every broker knows about every event. That would expend even more of scarce resouces, profusing the brokers and network with primitive events without demand. Effectively, it would lead to a duplication of the centralised broker in every one of the distributed brokers. Admittedly, all potential composite events would be found, but sadly, it would also practically guarantee congestion of the notification service as well as the underlying communication channels. Both solutions are clearly unacceptable.

Instead, the primitive events should be propagated through the right parts of the broker network without any client subscribing to them. That raises two questions: How should the propagation of events be controlled? And how far should they be allowed to spread through the notification service before they are dropped?

4.7.1 Subscription agents

A possible answer to the questions of how the propagation of events should be controlled, might be to introduce a sort of agent that subscribes to primitive events on behalf of the clients. Or more precisely: The agent should subscribe to low-level events that can be expected to give rise to composite events — which in turn are of interest to clients. Technically, the agent would function much like a client subscribing to low-level events that risk to be ignored. The major difference between an agent and a client is in location. The agent should reside together with one of the brokers in the network; either as an embedded component, or as a separate module (figure 4.13). Because peripheral connections tend to be more constrained than central ones, the

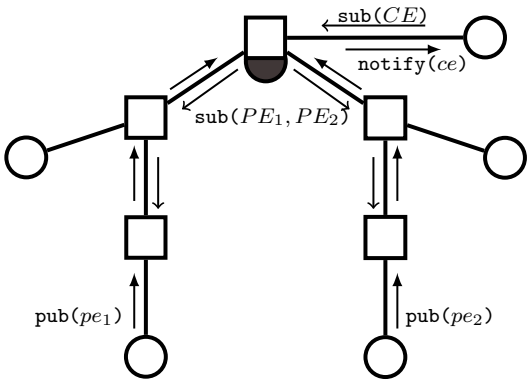


Figure 4.13: The centre top broker has an agent attached that subscribes to primitive events without demandd from any client.

agents should be located at central brokers in the part of the network that is most likely see the events in question. Each broker can have zero or more agents attached to it, each with a dedicated area of responsibility. The agent will subscribe to events that fall within its area of responsibility. For example, there could be an agent for traffic control in a certain area. That agent would subscribe to all kinds of events that might be relevant to the flow of traffic on (and off) roads in its area.

The agent will use the pub/sub service just like any ordinary client. The responsibility of the agent is limited to subscribing to the right kind of primitive events. Then, the broker can collect the incoming notifications as usual, and let composite events be discovered in the normal fashion — whatever that might be. With this design, separation of concern is observed, and the mechanisms of event propagation are used in a consistent manner. The agents would not take active steps to do anything with the incoming notifications. Thus, they are not really agents in the ordinary sense of the word. There could, of course, be agents of the active kind using the notification service as well, but then they would fall into the category of client in this context.

4.7.2 Advertisements

The second question; how far should event notifications be allowed to spread through the notification service before they are dropped, could be restated as: How far should *subscriptions* be allowed to spread before they are dropped? Events just track subscriptions back to the subscriber, so they are taken care of. With the design we have outlined so far, subscriptions are flooded through the network without constraint. Therefore, they will surely enter parts of the notification service where they are not likely to run into matching notifications. This will strain the underlying communications, and it will fill

up the routing tables with subscriptions without a purpose.

For example: Movement of neutral vehicles is interesting for traffic conditions locally, but hardly for anything else. So subscriptions to notifications about those should be disseminated to brokers connected to publisher which are located in the right area and can publish notifications about neutral vehicles. Outside of that, the subscription should be dropped. In contrast, movement of enemy vehicles will typically attract much wider interest, but still not universal.

What is needed is a mechanism that helps disseminate subscriptions as long as they are likely to encounter matching notifications, but no longer. In some pub/sub framework, this issue is adressed with *advertisements*[53]. An advertisements can be seen as a declaration of what kind of notifications a client is going to publish — a sort of a template. In the notification service, advertisements are essentially treated as subscriptions to subscriptions. So instead of flooding the subscriptions, the advertisements are flooded, and then the subscriptions are selectively routed back along the trace of matching advertisements.

With events represented as graphs, and subscriptions expressed as queries over graphs, advertisements would have to be represented in the same way as the events; as graphs. The difference would be that events notifications include all sorts of details about the specific event, whereas an advertisement will only include very general information applicable to all future events from the client that issues it. In a military setting, that would typically include event class, geographic area and organisational role and affiliation, amongst other.

For a broker to handle advertisements, a few changes to the architecture must be made. With advertisements working as subscriptions to subscriptions, a routing table for mapping advertisements to neighbour nodes must be in place. The handling of subscriptions will then become similar to the handling of notifications.

As for the message transmission between nodes, not much will change. The messages will still be wrappers, only now they may wrap advertisements in addition to subscriptions and notifications.

The intended advantage of advertisements is smaller routing tables, yielding faster matching in each broker. Then again, advertisements introduce another layer of overhead processing, which in itself incurs a cost: The advertisements will have to be flooded, brokers need to maintain separate routing tables of active advertisements, and the routing of subscriptions will require more processing than simple flooding. If the number of advertisements is more or less equal to the number of subscriptions, then not much will be gained. If, however, the number of advertisements is substantially lower, flooding will be reduced, and total traffic can be expected to diminish. Whether or not the total gain is greater than the cost, is an empirical question.

Chapter 5

Discussion

I have now presented my own ideas of what publish/subscribe and semantic technologies might have to contribute to the military domain, and to the realisation of Network Centric Warfare. Then, I have described a high-level design of a way to enhance publish/subscribe brokers with semantic technologies.

This chapter relates my ideas to some of the literature that treats similar questions. It also presents some issues that I have not treated in detail, but that needs to be considered for a future trial implementation.

5.1 Related work

In chapter 3, I briefly evaluated the suitability of the publish/subscribe model and semantic technologies for the military domain. This section will review some of the literature that discusses related questions. The purpose is to put my own ideas in perspective, and to see what expectations the scientific community has for future application of publish/subscribe and semantic technologies in the military domain.

Chapter 4 presents one possible way to enhance the capabilities of a publish/subscribe notification service with widely adopted semantic technologies. The final part of this section compares this approach with that of some other initiatives reported in the literature.

5.1.1 Publish/subscribe for the military domain

As far as I've been able to find, the publish/subscribe interaction model for military applications has not received a lot of attention as such. However, the qualities it has to offer are recognised as valuable to command and control systems, and especially in a network centric setting.

Potok et al.[61] note the need for loosely coupled and asynchronous message exchange as part of a networked command and control system. They

see this as a suitable communication and coordination mechanism between a collection of connected or disconnected agents. In particular, they refer to two coordination models; a blackboard-based model that provides a shared area where agents can send and receive messages, and a tuple-based model like the one in Linda[22]. The Linda model is based on message passing, where a message, or tuple, is a series of typed fields. Both models allow participating agents to be anonymous to one another, and fetch messages based on the message contents. They do not, however, include any parallell to the broker network and dissemination mechanism of publish/subscribe.

Gagnes[31] discusses the use of middleware for building the common operational picture on Network Based Defence. He points out a number of characteristics of military communications that the middleware must cope with; limited bandwidth, unreliable connectivity, high error rates and long delays. To meet this situation, some of the demands for middleware he identifies is that it make economical use of bandwidth and communicate asynchronously or by messaging.

Another class of middleware offering asynchronous, event driven message exchange is Web Services(WS). Somewhat confusingly, this is also called publish/subscribe even though it is not exactly the same concept as discussed in this thesis. The difference is that in 'pure' publish/subscribe, clients remain anonymous to one another, and communication is inherently one-to-many. In WS publish/subscribe, on the other hand, consumers register their subscriptions directly with the service provider, and not with the notification service. Thus, communication effectively becomes one-to-one.

Johnsen, Hafsø et al.[37, 45] explicitly discuss publish/subscribe in web services for use in Network Based Defence. "The asynchronous nature of the publish/subscribe paradigm makes it a very important mode of communications in NBD"[45, p. 22]. The asynchronous communication pattern is considered well suited to situations where informations is produced at irregular intervals. Publish/subscribe provides efficient information exchange, they say, delivering only what has been subscribed to, and thereby avoiding information overflow. Moreover, publish/subscribe is seen as a way to reduce network traffic. The last point, however, is considered an issue for possible improvement. Rasmussen et al.[63] conducted an experimental evaluation and observed that "The message distribution is a potential bottleneck since web services utilize point-to-point communication".

Johnsen, Hafsø et al. discuss the use of proxies to reduce the traffic volume. In general terms, a proxy server stands between two communicating parties. Used with WS publish/subscribe, it would stand between the service provider and the clients, performing a function that corresponds to a broker in 'pure' publish/subscribe. The clients will subscribe to the proxy, which forwards the first subscription of a particular type to the service provider. Then, the service provider will publish to the proxy, which in turn duplicates the message and notifies the consumers. With this extension, the WS

publish/subscribe concept looks very much like the concept described in section 3.1.

To summarise, publish/subscribe appears to meet important requirements of the future networked military information infrastructures. The very paradigm of publish/subscribe that I discuss may not have been referred to extensively, but its characteristics answer the need for loosely coupled, asynchronous and resilient message oriented information exchange.

5.1.2 Semantic technologies for the military domain

As mentioned in chapter 3, Forsvaret Forskningsinstitutt is investigating ways to make use of semantic technologies for military purposes. Their work is still in progress, but the initial report[39] concludes with saying that "Semantic technologies are a promising family of information technologies that potentially can deliver critical capabilities also in the military domain".

Smart et al.[69, 70] discuss the use of semantic technologies to enhance situational awareness. They particularly identify information retrieval, information sorting, information fusion, information dissemination knowledge processing and interaction and visualisation as interesting capability areas. This could, they believe, help decision makers in an heterogeneous and distributed environment get the information they need in line with task-variant information needs.

Lacy et al.[47] report about practical experiences with OWL in various experimental use cases in the military domain. One example is a decision support system for explosive ordnance disposal. Their lesson learnt is that for the right kinds of applications, OWL can be very useful.

Pulvermacher et al.[62] studied semantic linking across different military subdomains. Their chosen case was target validation, using data from different sources. Their approach was to make OWL ontologies with common higher-level concepts to link the different data sources. The main conclusions were that semantic linking is powerful, but complex, that better tools were needed (as of 2003), and that semantic technologies offer great potential for future applications.

The above are partly experimental, and partly theoretical evaluations of the applicability of semantic technologies to the military domain. From this, I conclude that semantic technologies are met with positive interest, and that my own evaluation in section3.4, is supported: It is worth investing effort in investigating semantic technologies further. It is, however, worth noting that this does not build on extensive practical experiments. The authors, especially Lacy and Pulvermacher, warn that working with ontologies is difficult, and that they are not necessarily a solution to all information problems.

5.1.3 Publish/subscribe with semantic filtering and routing

The limitations of syntactic filtering and routing in publish/subscribe notification services has been recognised and addressed by a number of researchers. One proposed solution, S-ToPSS[58], builds on an enhancement of conventional content-based attribute-value pairs to recognise synonyms and concept hierarchies. Antollini et al.[16, 15, 24] propose a *concept-based approach*. Their solution is to put a layer of common concept definitions between an underlying notification service and the client applications. Adapters transform and convert messages from clients to the shared vocabulary, and then back to the preferred format of the receiving client after the message has been distributed through the notification service.

Both the above solutions rely on customised formalisms for the specific solutions. Other approaches use more widely adopted formats and standards, most notably RDF.

One line of research addresses the problem of efficient dissemination of RDF Site Summary (RSS) feeds[60, 59, 1]. Their approach is similar to the one discussed here in that publications are represented as RDF graphs and subscriptions are represented as graph patterns. The syntactic expression of subscriptions has been a specially tailored adaptation of the RDF Query Language (RDQL)[67], which they call *G-ToPSS Query Language (GQL)*. It consists of five-tuples of the form *(subject, property, object, constraintSet(subject), constraintSet(object))* The main focus of their work is on efficient and scalable matching between the RDF graphs and the subscription graph pattern. In addition, the *G-ToPSS* system[60] performs matching based on class taxonomies. The work of Liu[1] adds *subscription containment* and *subscription merger*, which are two ways to optimise matching by exploiting commonalities between subscription graphs.

The *Ontology-based Publish/Subscribe (OPS) system*[80, 79] also uses RDF graphs to represent events and graph patterns to represent subscriptions. Efficient matching is a primary goal of this work as well, and a matching algorithm is described and evaluated. Again, a custom formalism for expressing subscriptions has been defined. The language is relatively intricate, and based on several existing RDF query languages. The structure is a subscription statement is a quadruple of (subject, object, meta-statement, [filter_func(object)]), where the meta-statement specifies type constraints.

The Semantic Publish/Subscribe System (SPS)[52] builds on the OPS, and is also motivated by the problem of efficient and scalable matching of RSS feeds. Matching semantics is extended by using OWL lite to describe the concepts involved in an event.

The three programmes above all use RDF graphs to express events, and centre their research on efficient and scalable matching. Performance is evidently important for any working implementation, especially when matching must be performed on internet scale. This is one important difference be-

tween their work and the approach of this thesis: I have paid little attention to performance of the matching process, instead depending on standards and available tools. However, their departure from the established standards by designing customised subscription languages makes the principles harder to implement and use in real life. None the less, performance is important, and their results are certainly interesting for my future work.

The Semantic Infosphere[77] is a prototype implementation building a semantic filtering capability on top of an existing publish/subscribe infrastructure. Interestingly, a military scenario is used for testing the prototype. Their approach was to augment standard message formats with semantic annotations using terms from a domain ontology. The formalism used to define the ontology is DAML+OIL[10], which is a predecessor to OWL. Subscriptions are expressed as DAML+OIL concept definitions, using the vocabulary from the domain ontology. The filtering amounts to performing instance classification with a FaCT reasoner[42] answer the question: "Is the DAML+OIL individual representing the message a member of the DAML+OIL class representing the subscription"?

The Semantic Message Oriented Middleware System of Li and Jiang[50] employs the same method: Publishers submit instances of a `topic` class in an DAML+OIL ontology, subscribers submit concept descriptions, and a match is found when an advertisement[1] satisfies the restrictions of a concept description.

The two projects based on DAML+OIL differ from the approach presented here in the way subscriptions are expressed and matching is performed. The difference between DAML+OIL and OWL is not fundamental, so the solutions proposed in [77] and [50] can easily be ported, should that be desirable.

To the extent that the approach outlined in chapter 4 presents anything original, it is the use of semantic technology to detect and construct composite events in the notification service. I have to the best of my ability not found the subject explicitly addressed in any of the work about enhancing publish/subscribe infrastructures with semantic filtering and routing.

5.2 Open issues

Chapter 4 outlined a high-level design of a publish/subscribe notification service. That description focused on the main concepts and ideas, passing by some issues that also should be addressed in any working implementation. Some of them can be regarded as problems that need to be handled, others as potential improvements to the basic design. Without going into great detail, this section will bring up some points that deserve attention, emphasising subjects that relate more or less directly to semantic technologies.

[1] They use the word *advertisement* instead of *notification*.

5.2.1 Information security and access control

As pointed out in section 2.2.5, security is an important issue for information systems used in a military setting. This is, of course, also true for most other domains and application scenarios, but military organisations are particularly sensitive on this point. As the discussion so far has been mostly about creating means for effective dissemination of information, the question of security has not been touched upon yet. None the less, the two are intimately connected: Giving and holding back access to information.

A publish/subscribe information infrastructure is by its very nature open and transparent. That is nothing short of the primary purpose; open information exchange between anonymous actors. In a typical implementation, anyone who has the right to issue subscriptions can subscribe to any kind of notification that is published through the NS — and receive the notifications. The underlying assumption is that all unauthorised access is prevented by other mechanisms; in the underlying network for instance. Once in, the entire service is open.

This situation is not satisfactory in a military setting. It is necessary to be able to discriminate between clients according to the rights they are granted in the real world. Access rights can be differentiated along multiple dimensions in the military domain. Security classifications, national restrictions, and authorisation based on organisational and functional role, are some of the most important. Information exchange needs to be open, but within bounds. At the same time, the loose coupling of publish/subscribe should be maintained, so the identity of notification consumers will remain unknown to the producers.

Once a broker has accepted a subscription from a client and passed it on to the next broker, the originating client is unknown: The second broker in the chain will only know the first broker, but not the client. The third broker will only know the second broker, and so on. Thus, the logical place to enforce access control is in the broker that receives a new subscription directly from a client. The security mechanism will function like a filter for new subscriptions. If a subscription is for information the client is not authorised to receive, it should be dropped. That way, the clients will receive only event notifications for which they are authorised.

To perform the filtering, the broker needs to have a model of information access control and an accompanying specification of the current policy. Next, the actor needs to be reliably authenticated each time it presents a new subscription. This is an entire field of research that I will not delve into here. Still, it should be mentioned that several approaches use semantic technologies and ontologies for this purpose[17, 25, 46, 28, 23]. Integrating this with a semantic notification service might be a viable approach.

5.2.2 Matching efficiency

Selective filtering is a way to reduce network traffic. However, if the filtering itself does not scale, the performance of the entire system will suffer. In general, more expressive mechanisms demand more processing and resources in the broker. At some point, the cost becomes too high. Even if event matching and routing is near perfect measured by the information content, delays of several minutes is hardly acceptable. Reasoning with semantic technologies is generally expensive. Thus, the right balance between loading the network and loading the brokers must be found.

One such balancing act presents itself in section 4.5.2. The question is which version of an event graph should be passed on to the next broker; the minimal one as received, or an extended one with the inferred entailments added? In a disadvantaged grid, it makes sense to keep messages as small as possible, which favours the minimal. That would, however, mean that a number of the entailments would have to be re-inferred by each broker the notification passes through. For example, the `FriendlySupplyConvoy` would need to be identified as an instance of `FriendlyConvoy` over and over again. Clearly, such duplication of work is a waste and should be avoided. In addition to these performance issues, there is a possibility of missing important entailments if only the minimal graph is forwarded. The reason is that brokers along the trace of a notification might hold somewhat different sets of background data, and the full set of possible entailments may depend on combining entailments from more than one of the involved brokers: Assuming
G is an event graph,
A_1 and A_2 are base assertions held in two different brokers,
E_1 and E_2 are entailments such that
$G \wedge A_1 \leftrightarrow R_1$ and
$G \wedge E_1 \wedge A_2 \leftrightarrow E_2$,
then E_2 will only be discovered if E_1 is included in the message with G. On the other hand: The only solution that can guarantee the discovery of every possible entailment is to collect all background data and all event graphs in one global knowledge base. That is clearly unfeasible, especially in a disadvantaged grid. So adopting a distributed solution like publish/subscribe necessarily means accepting a tradeoff between complete information and efficient information dissemination. Deciding the right balance is an engineering issue that must be considered carefully.

A task that will put a very heavy load on the brokers of any realistically sized imlementation, is spatial reasoning. Geographic information models are complex, and data volumes are massive. In my examples, I have dodged the problem by using 'quick and dirty' models that presumably do not scale. A working implementation can, for example, not contain the relative distance between any two points of interest in an operational area. Instead, more

general models must be in place. This is in itself a research field out of scope of this thesis[34, 41].

On a positive note: The volume of notifications passing through any one broker will hopefully be relatively manageable — at least compared to the volume semantic web applications might experience. First, only a portion of all traffic in a military setting is best disseminated through a publish/subscribe infrastructure. One-to-many messages exchanged between anonymous clients. Communication that is inherently one-to-one, where the recipient is — or can become known — should be transmitted by other means. Second, there will probably be relatively few clients connected to each broker. Military organisations are large, but not as large as the world wide web, where millions might access the same service. Third, a disadvantaged grid will effectively set a limit to the information flow. In all, these factors are favorable to the proposed solution. Even so, optimisations should be made wherever possible, and the structuring of subscriptions is one area of improvement.

5.2.3 Subscription covering and merging

The design of a broker suggested in section 4.5 is in no way optimised for performance or scalability. In particular, the routing table is inefficiently organised: Every single subscription query is run against every incoming event graph. The flat structure of the routing table does not take advantage of similarities and partial overlaps between subscriptions to reduce the number of matching attempts. Take, for example, one query that includes the clause `(?actor belongsTo coyC)`. If that clause cannot be satisfied by the event graph, all other queries that include the same non-optional clause can be known not to match. Therefore, they can all be written off the same instant. With a flat table, however, all queries are treated as being entirely dissimilar as long as there is at least one distinguishing feature between them, and each will in turn be tried against the same event graph.

Methods for exploiting similarities between subscriptions have been formalised and described in [53]. `Identity-based routing` is a relative simple optimisation of identical subscriptions. The next step is *covering-based routing*. One subscription is said to *cover* another if it matches all notifications that are matched by the other. This is similar to a subset–superset relation. *Merging-based routing* is based on creating a new subscription that covers two or more 'narrower' subscriptions. The merged subscriptions is then forwarded instead of the original ones. These oprimisations contribute to reduced exchange of subscriptions between the brokers as well as more efficient matching within each broker. A similar optimisation approach for RDF-like graph patterns is described in [1]. Even though their proposed subscription language is expressed as sets of five-tuples, and as such quite different from SPARQL, the oprimisations they propose should in principle

be applicable to the resulting graph patterns.

Chapter 6

Conclusions

6.1 Summary

To summarise the main points of the thesis, I will turn back to the problem statements from the introduction:

1. How can information routing and event composition in a publish/subscribe infrastructure be enhanced with semantic technologies?

2. How can the two technologies together be used for selective information dissemination in a network centric military force?

The architectural outline of a notification broker with semantic matching and event composition capabilities, indicates that semantic technologies and publish/subscribe in combination should be able to provide a powerful notification service. Without an experimental implementation, however, it is a bit early to give a definitive and reliable answer to the positive. In the course of my work, I have come across complications and difficulties that need to be addressed, but that I have not had the opportunity to cover in full depth. The two most important areas, as mentioned in chapter 5, are security and efficiency. Security is critically important, and as noted, mechanisms for security management are available or under development. Efficiency is also important, but more a question of degree. Algorithms and tools within semantic technologies are under constant development, and will hopefully come to good use.

Semantic technologies and publish/subscribe go well together in the sense that they both are suited for information sharing in a distributed and heterogeneous environment. They answer some of the essential requirements of the military domain in general, and Network Centric Warfare in particular. Pub/sub is the more established of the two, and from my studies, I find that it offers the kind of decoupled information exchange that a flexible and networked organisation needs. In the scientific community, semantic technologies have been more widely studied and evaluated with military

application in mind. In line with my own view, the findings are that they look promising, but they have yet to prove their worth in real life.

Through arguments and examples, I have demonstrated ways that the proposed solution can be used for selective information dissemination in a network centric military force. This solution certainly cannot be expected to meet every need for information exchange, so the right scenarios and use cases must be identified. Event-driven, one-to-many messaging, and information exchange where producers and consumers are anonymous to one another, are the kinds of interactions this approach supports. Thus, it might deserve a place as one of several elements in the information infrastructure.

6.2 Future work

The obvious next step is a practical evaluation of the outlined solution. That means building an experimental implementation and running it through simulations and — if possible — practical experiments.

Three main components of such an implementation and evaluation need to be in place: The first is the notification broker in the form of a software component. The second is the set of ontologies and rules for the knowledge base, and for definition of subscriptions and notifications. The third is a test environment. This includes realistic and representative test cases with corresponding test data.

From what I have experienced, all these components present conciderable challenges.

Bibliography

[1] Efficient Filtering of RSS Documents on Computer Cluster.

[2] Free On-Line Dictionary of Computing. http://foldoc.org/.

[3] Notation 3. http://www.w3.org/DesignIssues/Notation3.html.

[4] The protégé ontology editor and knowledge acquisition system. TheProt{\'e}g{\'e}OntologyEditorandKnowledgeAcquisitionSystem.

[5] RDF/XML syntax specification. http://www.w3.org/TR/rdf-syntax-grammar/.

[6] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[7] SWRLTab builtin libraries. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries.

[8] Topicmaps.org. http://www.topicmaps.org/.

[9] Marine Corps Doctrinal Publication 6, Command and Control. http://www.au.af.mil/au/awc/awcgate/mcdp6/toc.htm, 1996.

[10] DAML+OIL (march 2001) reference description. http://www.w3.org/TR/daml+oil-reference, 2001.

[11] W3C Semantic Web Activity. http://www.w3.org/2001/sw/, 2009.

[12] David S. Alberts, John J. Garstka, and F.P. Stein and. *Network centric warfare: Developing and leveraging information superiority*. United States Dept. of Defense C41SR Cooperative Research Program, 2nd edition, 1999.

[13] David S. Alberts, John J. Garstka, Richard E. Hayes, and David A. Signori. *Understanding Information Age Warfare*. US DoD Command and Control Research Program (CCRP), 2001.

[14] David S. Alberts and Richard E. Hayes. *Power to the Edge: Command and Control in the Information Age*. Information Age Transformation Series. US DoD Command and Control Research Program (CCRP), June 2003.

[15] J. Antollini, Mario Antollini, P. Guerrero, and Mariano Cilia. Extending Rebeca to Support Concept-based Addressing. In *First Argentine Symposium on Information Systems (ASIS 2004), Sept*, volume 177, page 178, 2004.

[16] Mario Antollini, Mariano Cilia, and Alejandro P. Buchmann. Implementing a high level pub/sub layer for enterprise information systems. In Yannis Manolopoulos, Joaquim Filipe, Panos Constantopoulos, and José Cordeiro, editors, *ICEIS (1)*, pages 54–62, 2006.

[17] R. Ashri, T. Payne, D. Marvin, M. Surridge, and S. Taylor. Towards a semantic web security infrastructure. *Proc. of Semantic Web Services*, 2004.

[18] Roberto Baldoni and Antonio Virgillito. Distributed event routing in publish/subscribe communication systems: a survey. Technical report, Dipartimento di Informatica e Sistemistica, Università di Roma 'La Sapienza', 2005.

[19] Allen W. Batschelet. Effects-based operations: A new operational model?, 2002.

[20] Toni A. Bishop and Ramesh K. Karne. A Survey of Middleware. In *Proc. of 18th Intl. Conference on Computers and their applications*, pages 254–258, 2003.

[21] Chris Bizer. Triql - a query language for named graphs. `http://www4.wiwiss.fu-berlin.de/bizer/TriQL/`, 2004.

[22] Nicholas Carriero and David Gelernter. Linda in context. *Commun. ACM*, 32(4):444–458, 1989.

[23] Mohammad M. R. Chowdhury and Josef Noll. A distributed identity handling approach enriched with identity semantics. *IARIA Journal*, 1(1), 2009.

[24] M. Cilia, M. Antollini, C. Bornhovd, and A. Buchmann. Dealing with heterogeneous data in pub/sub systems: The Concept-Based approach. In *Intl Workshop on Distributed Event-Based Systems (DEBS'04)*, 2004.

[25] G. Denker, L. Kagal, and T. Finin. Security in the Semantic Web using OWL. *Information Security Technical Report*, 10(1):51–58, 2005.

[26] Mica R. Endsley. Design and evaluation for situation awareness enhancement. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 32, pages 97–101. Human Factors and Ergonomics Society, 1988.

[27] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec. The many faces of publish/subscribe. *ACM computing Surveys*, 35(2):114–131, 2003.

[28] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. ROWLBAC: representing role based access control in OWL. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 73–82. ACM New York, NY, USA, 2008.

[29] Forsvarets Stabsskole. *Forsvarets fellesoperative doktrine*. Forsvarsstaben, 2007.

[30] Forsvarsdepartementet. Forsvarssjefens forsvarsstudie 2007, 2007.

[31] Tommy Gagnes and Kjell Rose. Bruk av mellomvare i demonstrator for bildeoppbygging. Technical report, Forsvarets Forskningsinstitutt, 2003.

[32] Tim Grant and Bas Kooter. Comparing OODA & other models as Operational View C2 Architecture, 2005.

[33] Marco Grobelnik and Dunja Mladenić. Knowledge discovery for ontology construction. In John Davies, Rudi Studer, and Paul Warren, editors, *Semantic Web Technologies*, pages 9–27. John Wiley and Sons, 2006.

[34] R. Grütter and B. Bauer-Messmer. Towards spatial reasoning in the semantic web: A hybrid knowledge representation system architecture. *Lecture Notes in Geoinformation and Cartography*, 2007.

[35] Volker Haarslev and Ralf Möller. Incremental query answering for implementing document retrieval services. In *Proceedings of the International Workshop on Description Logics*, pages 85–94, 2003.

[36] Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz. A comparison of RDF query languages. In *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004.*, NOV 2004.

[37] Trude Hafsøe, Frank T Johnsen, Ketil Lund, and Anders Eggen. Adapting web services for limited bandwidth tactical networks, published at the 12th iccrts. Technical report, Forsvarets Forskningsinstitutt, 2007.

[38] DL Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[39] Bjørn Jervell Hansen, Tommy Gagnes, Rolf Rasmussen, Marianne Rustad, and Geir Sletten. Semantic technologies. Technical report, Forsvarets Forskningsinstitutt, 2007.

[40] Ole-Erik Hedenstad. Informasjonsinfrastruktur for NBF. Technical report, Forsvarets Forskningsinstitutt, 2002.

[41] K. Hiramatsu and F. Reitsma. GeoReferencing the Semantic Web: ontology based markup of geographically referenced information. In *Joint EuroSDR/EuroGeographics workshop on Ontologies and Schema Translation Services*, 2004.

[42] Ian Horrocks. FaCT and iFaCT. In *Proceedings of the International Workshop on Description Logics (DL'99)*, volume 22, pages 133–135, 1999.

[43] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml. `http://www.w3.org/Submission/SWRL/`, 2004.

[44] Kevin Hutt. A Comparison of RDF Query Languages. In *Proc. of 21th Computer Science Seminar, Hartfort, Connecticut*, pages 1–7, 2005.

[45] Frank Trethan Johnsen, Trude Hafsøe, and Ketil Lund. Quality of service considerations for network based defence. Technical report, Forsvarets Forskningsinstitutt, 2007.

[46] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Using semantic web technologies for policy management on the web. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 1337. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[47] L. Lacy, G. Aviles, K. Fraser, W. Gerber, A. Mulvehill, and R. Gaskill. Experiences using OWL in military applications. In *Proc. of the First OWL Experiences and Directions Workshop*, volume 188, 2005.

[48] Guoli Li and H.A. Jacobsen. Composite subscriptions in content-based publish/subscribe systems. In *Middleware 2005*, volume 3790 of *Lecture notes in computer science*, pages 249–269. Springer, 2005.

[49] Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. Adaptive content-based routing in general overlay topologies. In *Middleware 2008*, volume 5346/2008 of *Lecture notes in computer science*, pages 1–21. Springer, 2008.

[50] Han Li and Guofei Jiang. Semantic message oriented middleware for publish/subscribe networks. In Edward M. Carapezza, editor, *PROCEEDINGS SPIE*, volume 5403, pages 124–133. International Society for Optical Engineering, SPIE, 2004.

[51] Ying Liu and Beth Plale. Survey of publish subscribe event systems. Technical report, Computer Science Department, Indiana University, 2003.

[52] JianGang Ma, Gang Xu, JinLing Wang, and Tao Huang. A semantic publish/subscribe system for selective dissemination of the rss documents. *International Conference on Grid and Cooperative Computing*, pages 432–439, 2006.

[53] Gero Mühl, Ludger Fiege, and Peter Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag New York, Secaucus, NJ, USA, 2006.

[54] Eduardo F. Nakamura, Antonio A.F. Loureiro, and Alejandro C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, 39(3), 2007.

[55] NATO Research and Technology Organisation. Validation and verification of NATO network enabled capabilities, 2008.

[56] Z. Pan, A. Qasem, and J. Heflin. An investigation into the feasibility of the semantic web. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 1394. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[57] Steve Pepper. The TAO of topic maps. In *Proceedings of XML Europe*, 2000.

[58] Milenko Petrovic, Ioana Burcea, and Hans-Arno Jacobsen. S-topss: semantic toronto publish/subscribe system. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 1101–1104. VLDB Endowment, 2003.

[59] Milenko Petrovic, Haifeng Liu, and Hans-Arno Jacobsen. Cms-topss: efficient dissemination of rss documents. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1279–1282. VLDB Endowment, 2005.

[60] Milenko Petrovic, Haifeng Liu, and Hans-Arno Jacobsen. G-topss: fast filtering of graph-based metadata. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 539–547, New York, NY, USA, 2005. ACM.

[61] T. Potok, L. Phillips, R. Pollock, and A. Loebl. Suitability of agent technology for military command and control in the future combat system environment, 2003.

[62] M.K. Pulvermacher, S. Stoutenburg, S. Semy, and S.P. Staff. Netcentric Semantic Linking. Technical report, MITRE Technical Report, 2004.

[63] Rolf Rasmussen, Anders Eggen, Dinko Hadzic, Ole-Erik Hedenstad, Raymond Haakseth, and Ketil Lund. Experiment report: "Secure SOA supporting NEC" - NATO CWID 2006. Technical report, Forsvarets Forskningsinstitutt, 2006.

[64] Dave Reynolds, Carol Thompson, Jishnu Mukerji, and Derek Coleman. An assessment of RDF/OWL modelling. Technical report, Hewlett Packard Labs, 2005.

[65] Peter Rysavy. Data capabilities: GPRS to HSDPA and beyond. Whitepaper, Rysavy Research, 2005.

[66] Douglas C. Schmidt, Richard E. Schantz, Michael W. Masters, Joseph K. Cross, David C. Sharp, and Louis P. DiPalma. Towards Adaptive and Reflective Middleware for Network-Centric Combat Systems. *CrossTalk*, 2001.

[67] Andy Seaborne. Rdql - a query language for rdf. `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`, 2004.

[68] Nigel Shadbolt, Wendy Hall, and Tim Berners-Lee. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[69] P. R. Smart, N.R. Shadbolt, L.A. Carr, and M.C. Schraefel. Knowledge-based information fusion for improved situational awareness. In *Information Fusion, 2005 8th International Conference on*, volume 2, 2005.

[70] P.R. Smart, A. Bahrami, D. Braines, D. McRae-Spencer, J. Yuan, and N.R. Shadbolt. Semantic Technologies and Enhanced Situation Awareness. In *1st Annual Conference of the International Technology Alliance (ACITA), Maryland, USA*, 2007.

[71] Edward A. Smith. Effects Based Operations. *Security Challenges*, 2(1):60, 2006.

[72] Diane H. Sonnenwald, Kelly L. Maglaughlin, and Mary C. Whitton. Designing to support situation awareness across distances: an example from a scientific collaboratory. *Information Processing and Management*, 40(6):989–1011, 2004.

[73] Andrew S. Tanenbaum and Marten Van Steen. *Distributed systems*. Prentice Hall Upper Saddle River, NJ, 2002.

[74] S. Tarkoma and K. Raatikainen. State of the Art Review of Distributed Event Systems. Technical report, Helsinki University Computer Science Department, 2005.

[75] T.Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[76] UK Ministry of Defence. Network enabled capability, 2005.

[77] Michael Uschold, F. Dickey, C. Fung, S. Smith, S. Uczekaj, M. Wilke, S. Bechhofer, and I. Horrocks. A Semantic Infosphere. In *The Semantic Web, ISWC 2003: Second International Semantic Web Conference*, pages 882–896. Springer, 2003.

[78] B. J. A. van Bezooijen, P. J. M. D. Essens, and A. L. W. Vogelaar. Military self-synchronization: An exploration of the concept. In *11th ICCRTS, Coalition Command and Control in the Networked Era*, 2006.

[79] J. Wang, B. Jin, J. Li, and D. Shao. A semantic-aware publish/subscribe system with RDF patterns. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 141–146, 2004.

[80] Jinling Wang, Beihong Jin, and Jing Li. An ontology-based publish/subscribe system. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 232–253, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[81] Leoni Warne, Irena Ali, Derek Bopping, Dennis Hart, and Celina Pascoe. *The Network Centric Warrior: the Human Dimension of Network Centric Warfare*. Defence Science and Technology Organisation, Information Sciences Laboratory, 2004.